# SIMATIC S5

# S5-100U
# Programmable Controller

**System Manual**

**CPU 100/102/103**

**EWA 4NEB 812 6120-02b**

**Edition 04**

www.rgbautomatyka.pl

STEP ® SINEC ® and SIMATIC ® are registered trademarks of Siemens AG.
LINESTRA® is a registered trademark of the OSRAM Company.
Subject to change without prior notice.

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Copyright© **Siemens AG 1992**

*EWA 4NEB 812 6120-02b*

Introduction

The SIMATIC S5 System Family 1

Technical Description 2

Installation Guidelines 3

Start-Up and Program Tests 4

Diagnostics and Troubleshooting 5

Addressing 6

Introduction to STEP 5 7

STEP 5 Operations 8

Integrated Blocks and Their Functions 9

Interrupt Processing 10

Analog Value Processing 11

The Integral Real-Time Clock, for CPU 103 and Higher 12

Connecting the S5-100U to SINEC L1 13

Module Spectrum 14

Function Modules 15

Appendices A/B/C D/E/F

Index

*EWA 4NEB 812 6120-02b*

# Contents


| | | Page |
|---|---|---|
| **How to Use This Manual** | | **xv** |
| **1** | **The SIMATIC S5 System Family** | **1 - 1** |
| **2** | **Technical Description** | **2 - 1** |
| 2.1 | Programmable Controller Design | 2 - 1 |
| 2.2 | Principle of Operation for the Programmable Controller | 2 - 3 |
| 2.2.1 | Functional Units | 2 - 3 |
| 2.2.2 | Mode of Operation for the External I/O Bus | 2 - 6 |
| **3** | **Installation Guidelines** | **3 - 1** |
| 3.1 | Installing S5-100U Components | 3 - 1 |
| 3.1.1 | Assembling a Tier | 3 - 1 |
| 3.1.2 | Multi-Tier Expansion | 3 - 5 |
| 3.1.3 | Cabinet Mounting | 3 - 7 |
| 3.1.4 | Vertical Mounting | 3 - 8 |
| 3.2 | Wiring | 3 - 9 |
| 3.2.1 | Connection Methods: Screw-Type Terminals and Crimp Snap-in | 3 - 9 |
| 3.2.2 | Connecting the Power Supply to the S5-100U | 3 - 12 |
| 3.2.3 | Connecting Digital Modules | 3 - 13 |
| 3.2.4 | Connecting the Digital Input/Output Module | 3 - 18 |
| 3.3 | Electrical Configuration | 3 - 20 |
| 3.3.1 | Electrical Configuration for the S5-100U | 3 - 20 |
| 3.3.2 | Electrical Configuration with External I/Os | 3 - 21 |
| 3.3.3 | Non-Floating and Floating Configurations | 3 - 25 |
| 3.4 | Wiring Arrangement, Shielding, and Measures to Guard against Electromagnetic Interference | 3 - 29 |
| 3.4.1 | Running Cables Inside and Outside a Cabinet | 3 - 29 |
| 3.4.2 | Running Cables Outside Buildings | 3 - 30 |
| 3.4.3 | Equipotential Bonding | 3 - 31 |
| 3.4.4 | Shielding Cables | 3 - 32 |
| 3.4.5 | Special Measures for Interference-Free Operation | 3 - 33 |

| | | Page |
|---|---|---|
| **4** | **Start-Up and Program Tests** | **4 - 1** |
| 4.1 | Operating Instructions | 4 - 1 |
| 4.1.1 | CPU Operator Panel | 4 - 1 |
| 4.1.2 | Operating Modes | 4 - 1 |
| 4.1.3 | Performing an Overall Reset on the Programmable Controller | 4 - 2 |
| 4.2 | Starting Up a System | 4 - 3 |
| 4.2.1 | Suggestions for Configuring and Installing the Product | 4 - 3 |
| 4.2.2 | Procedures for Starting Up the Programmable Controller | 4 - 4 |
| 4.3 | Loading the Program into the Programmable Controller | 4 - 5 |
| 4.4 | Backing Up the Program | 4 - 7 |
| 4.4.1 | Backing Up the Program on a Memory Submodule | 4 - 7 |
| 4.4.2 | Function of the Back-Up Battery | 4 - 8 |
| 4.5 | Program-Dependent Signal Status Display “STATUS” | 4 - 8 |
| 4.6 | Direct Signal Status Display “STATUS VAR” | 4 - 9 |
| 4.7 | Forcing Outputs, “FORCE”, for CPU 103 and Higher | 4 - 10 |
| 4.8 | Forcing Variables, “FORCE VAR” | 4 - 10 |
| 4.9 | Search Function | 4 - 11 |
| 4.10 | Program Check, for CPU 103 and Higher | 4 - 11 |
| **5** | **Diagnostics and Troubleshooting** | **5 - 1** |
| 5.1 | Indication of Errors by LEDs | 5 - 1 |
| 5.2 | CPU Malfunctions | 5 - 1 |
| 5.2.1 | “ISTACK” Analysis Function | 5 - 1 |
| 5.2.2 | Interrupt Analysis | 5 - 4 |
| 5.2.3 | Errors during Program Copying | 5 - 5 |
| 5.2.4 | Explanation of the Mnemonics Used in “ISTACK” | 5 - 6 |
| 5.3 | Program Errors | 5 - 8 |
| 5.3.1 | Locating the Error Address | 5 - 8 |
| 5.3.2 | Tracing the Program with the “BSTACK” Function | 5 - 11 |
| 5.4 | I/O Faults | 5 - 12 |
| 5.5 | System Parameters | 5 - 12 |
| 5.6 | The Last Resort | 5 - 13 |

| | | Page |
|---|---|---|
| **6** | **Addressing** | **6 - 1** |
| 6.1 | Slot Numbering | 6 - 1 |
| 6.2 | Digital Modules | 6 - 4 |
| 6.3 | Analog Modules | 6 - 5 |
| 6.4 | Combined Input Modules and Output Modules | 6 - 6 |
| 6.4.1 | Output Modules with Error Diagnostics | 6 - 6 |
| 6.4.2 | Digital Input/Output Module, 16 Inputs, 16 Outputs, 24 V DC for All CPUs Version 8MA02 and Higher and for CPU 102, Version 8MA01, Revision 5 and Higher | 6 - 7 |
| 6.4.3 | Function Modules | 6 - 7 |
| 6.5 | The Structure of Process Image Input and Output Tables | 6 - 8 |
| 6.5.1 | Accessing the Process Image Input Table (PII) | 6 - 10 |
| 6.5.2 | Accessing the Process Image Output Table (PIQ) | 6 - 11 |
| 6.6 | Interrupt Process Images Tables and Time-Controlled Program Processing in OB13 for CPU 103, Version 8MA02 and Higher | 6 - 12 |
| 6.6.1 | Accessing the Interrupt PII | 6 - 12 |
| 6.6.2 | Accessing the Interrupt PIQ | 6 - 14 |
| 6.7 | RAM Address Assignments | 6 - 15 |
| **7** | **Introduction to STEP 5** | **7 - 1** |
| 7.1 | Writing a Program | 7 - 1 |
| 7.1.1 | Methods of Representation | 7 - 1 |
| 7.1.2 | Operand Areas | 7 - 3 |
| 7.1.3 | Circuit Diagram Conversion | 7 - 3 |
| 7.2 | Program Structure | 7 - 4 |
| 7.2.1 | Linear Programming | 7 - 4 |
| 7.2.2 | Structured Programming | 7 - 5 |
| 7.3 | Block Types | 7 - 7 |
| 7.3.1 | Organization Blocks | 7 - 9 |
| 7.3.2 | Program Blocks | 7 - 11 |
| 7.3.3 | Sequence Blocks, for CPU 103 and Higher | 7 - 11 |
| 7.3.4 | Function Blocks | 7 - 11 |
| 7.3.5 | Data Blocks | 7 - 16 |
| 7.4 | Program Processing | 7 - 18 |
| 7.4.1 | Program Processing with CPU 102 | 7 - 19 |
| 7.4.2 | START-UP Program Processing | 7 - 24 |
| 7.4.3 | Cyclic Program Processing | 7 - 26 |
| 7.4.4 | Time-Controlled Program Processing, for CPU 103 Version 8MA02 and Higher | 7 - 28 |

Page

7.4.5 Interrupt-Driven Program Processing, for CPU 103 Version 8MA02 and Higher . . . . . 7 - 29

7.5 Processing Blocks . . . . . 7 - 30
7.5.1 Changing Programs . . . . . 7 - 30
7.5.2 Changing Blocks . . . . . 7 - 30
7.5.3 Compressing the Program Memory . . . . . 7 - 30

7.6 Number Representation . . . . . 7 - 31

**8 STEP 5 Operations . . . . . 8 - 1**

8.1 Basic Operations . . . . . 8 - 1
8.1.1 Boolean Logic Operations . . . . . 8 - 2
8.1.2 Set/Reset Operations . . . . . 8 - 7
8.1.3 Load and Transfer Operations . . . . . 8 - 10
8.1.4 Timer Operations . . . . . 8 - 15
8.1.5 Counter Operations . . . . . 8 - 25
8.1.6 Comparison Operations . . . . . 8 - 30
8.1.7 Arithmetic Operations . . . . . 8 - 31
8.1.8 Block Call Operations . . . . . 8 - 33
8.1.9 Other Operations . . . . . 8 - 38

8.2 Supplementary Operations . . . . . 8 - 39
8.2.1 Load Operation, for CPU 103 and Higher . . . . . 8 - 40
8.2.2 Enable Operation, for CPU 103 and Higher . . . . . 8 - 41
8.2.3 Bit Test Operations, for CPU 103 and Higher . . . . . 8 - 42
8.2.4 Digital Logic Operations . . . . . 8 - 44
8.2.5 Shift Operations . . . . . 8 - 48
8.2.6 Conversion Operations . . . . . 8 - 50
8.2.7 Decrement/Increment, for CPU 103 and Higher . . . . . 8 - 52
8.2.8 Disable/Enable Interrupt, for CPU 103 Version 8MA02 and Higher . . . . . 8 - 53
8.2.9 “DO” Operation, for CPU 103 and Higher . . . . . 8 - 54
8.2.10 Jump Operations . . . . . 8 - 56
8.2.11 Substitution Operations, for CPU 103 and Higher . . . . . 8 - 58

8.3 System Operations, for CPU 103 and Higher . . . . . 8 - 64
8.3.1 Set Operations . . . . . 8 - 64
8.3.2 Load and Transfer Operations . . . . . 8 - 64
8.3.3 Arithmetic Operations . . . . . 8 - 67
8.3.4 Other Operations . . . . . 8 - 68

8.4 Condition Code Generation . . . . . 8 - 69

8.5 Sample Programs . . . . . 8 - 71
8.5.1 Momentary-Contact Relay/Edge Evaluation . . . . . 8 - 71
8.5.2 Binary Scaler/Binary Divider . . . . . 8 - 71
8.5.3 Clock/Clock-Pulse Generator . . . . . 8 - 73

| | | Page |
|---|---|---|
| **9** | **Integrated Blocks and Their Functions** | **9 - 1** |
| 9.1 | Assigning Internal Functions to DB1, for CPU 103 Version 8MA03 and Higher | 9 - 1 |
| 9.1.1 | Configuration and Default Settings for DB1 | 9 - 1 |
| 9.1.2 | Setting the Address for the Parameter Error Code in DB1 | 9 - 2 |
| 9.1.3 | Assigning Parameters in DB1 | 9 - 4 |
| 9.1.4 | Rules for Setting Parameters in DB1 | 9 - 4 |
| 9.1.5 | How to Recognize and Correct Parameter Errors | 9 - 6 |
| 9.1.6 | Transferring DB1 Parameters to the Programmable Controller | 9 - 9 |
| 9.1.7 | Reference Guide for Setting Parameters in DB1 | 9 - 10 |
| 9.1.8 | Defining System Characteristics in DB1 | 9 - 11 |
| 9.2 | Integrated Function Blocks, for CPU 102 Version 8MA02 and Higher | 9 - 11 |
| 9.2.1 | Code Converter : B4 - FB240 - | 9 - 12 |
| 9.2.2 | Code Converter : 16 - FB241 - | 9 - 12 |
| 9.2.3 | Multiplier : 16 - FB242 - | 9 - 13 |
| 9.2.4 | Divider : 16 - FB243 - | 9 - 13 |
| 9.2.5 | Analog Value Conditioning Modules FB250 and FB251 | 9 - 14 |
| 9.3 | Integrated Organization Blocks | 9 - 14 |
| 9.3.1 | Scan Time Triggering OB31, for CPU 103 and Higher | 9 - 14 |
| 9.3.2 | Battery Failure OB34 | 9 - 14 |
| 9.3.3 | OB251 PID Algorithm, for CPU 103 Version 8MA02 and Higher | 9 - 15 |
| **10** | **Interrupt Processing** | **10 - 1** |
| 10.1 | Interrupt Processing with OB2, for CPU 103 Version 8MA02 and Higher | 10 - 1 |
| 10.2 | Calculating Interrupt Reaction Times | 10 - 5 |
| **11** | **Analog Value Processing** | **11 - 1** |
| 11.1 | Analog Input Modules | 11 - 1 |
| 11.2 | Connecting Current and Voltage Sensors to Analog Input Modules | 11 - 1 |
| 11.2.1 | Voltage Measurement with Isolated or Non-Isolated Thermocouples | 11 - 2 |
| 11.2.2 | Two-Wire Connection of Voltage Sensors | 11 - 3 |
| 11.2.3 | Two-Wire Connection of Current Sensors | 11 - 4 |
| 11.2.4 | Connection of Two-Wire and Four-Wire Transducers | 11 - 4 |
| 11.2.5 | Connection of Resistance Thermometers | 11 - 6 |
| 11.3 | Start-Up of Analog Input Modules | 11 - 7 |
| 11.4 | Analog Value Representation of Analog Input Modules | 11 - 11 |

Page


11.5 Analog Output Modules . . . . . 11 - 19
11.5.1 Connection of Loads to Analog Output Modules . . . . . 11 - 19
11.5.2 Analog Value Representation of Analog Output Modules . . . . . 11 - 20

11.6 Analog Value Conversion: Function Blocks FB250 and FB251 . . . . . 11 - 22
11.6.1 Reading in and Scaling an Analog Value - FB250 - . . . . . 11 - 22
11.6.2 Outputting of Analog Values - FB251 - . . . . . 11 - 25

**12 The Integral Real-Time Clock, for CPU 103 Version 8MA02 and Higher . . . 12 - 1**

12.1 Function . . . . . 12 - 1

12.2 Setting Parameters in DB1, for CPU 103 Version 8MA03 and Higher . . . . . 12 - 2
12.2.1 Defaults . . . . . 12 - 2
12.2.2 Reading the Current Clock Time and the Current Date . . . . . 12 - 3
12.2.3 DB1 Parameters Used for the Integral Real-Time Clock . . . . . 12 - 4

12.3 Programming the Integral Real-Time Clock in DB1, for CPU 103 Version 8MA03 and Higher . . . . . 12 - 5
12.3.1 Setting the Clock in DB1 . . . . . 12 - 5
12.3.2 Setting the Prompt Time in DB1 . . . . . 12 - 6
12.3.3 Setting the Operating Hours Counter in DB1 . . . . . 12 - 7
12.3.4 Entering the Clock Time Correction Factor in DB1 . . . . . 12 - 7

12.4 Structure of the Clock Data Area . . . . . 12 - 8

12.5 Structure of the Status Word and How to Scan It . . . . . 12 - 12

12.6 Setting Parameters for the Clock Data Area and the Status Word in the System Data Area . . . . . 12 - 15

12.7 Programming the Integral Real-Time Clock in the User Program . . . . . 12 - 21
12.7.1 Reading and Setting the Clock . . . . . 12 - 21
12.7.2 Programming the Prompt Function . . . . . 12 - 25
12.7.3 Programming the Operating Hours Counter . . . . . 12 - 30
12.7.4 Entering the Clock Time Correction Factor . . . . . 12 - 35

| | | Page |
|---|---|---|
| **13** | **Connecting the S5-100U to SINEC L1, for CPU 102 and Higher** | **13 - 1** |
| 13.1 | Connecting the Programmable Controllers to the L1 Bus Cable | 13 - 1 |
| 13.2 | Setting Parameters in the Programmable Controller for Exchanging Data | 13 - 1 |
| 13.2.1 | How to Program in a Function Block, for CPU 102 and Higher | 13 - 2 |
| 13.2.2 | Setting Parameters in DB1, for CPU 103 and Higher | 13 - 5 |
| 13.3 | Coordinating Data Exchange in the Control Program | 13 - 7 |
| 13.3.1 | Sending Data | 13 - 8 |
| 13.3.2 | Receiving Data | 13 - 9 |
| 13.3.3 | Programming the Messages in a Function Block | 13 - 11 |
| **14** | **Module Spectrum** | **14 - 1** |
| 14.1 | General Technical Specifications | 14 - 3 |
| 14.2 | Power Supply Modules | 14 - 4 |
| 14.3 | Central Processing Units | 14 - 7 |
| 14.4 | Bus Units | 14 - 10 |
| 14.5 | Interface Modules | 14 - 14 |
| 14.6 | Digital Modules | 14 - 16 |
| 14.6.1 | Digital Input Modules | 14 - 16 |
| 14.6.2 | Digital Output Modules | 14 - 26 |
| 14.6.3 | Digital Input/Output Modules | 14 - 36 |
| 14.7 | Analog Modules | 14 - 38 |
| 14.7.1 | Analog Input Modules | 14 - 38 |
| 14.7.2 | Analog Output Modules | 14 - 56 |
| **15** | **Function Modules** | **15 - 1** |
| 15.1 | Comparator Module 2×1 to 20 mA/0.5 to 10 V | 15 - 1 |
| 15.2 | Timer Module 2×0.3 to 300 s | 15 - 4 |
| 15.3 | Simulator Module | 15 - 7 |
| 15.4 | Diagnostic Module | 15 - 9 |

**Page**


15.5 Counter Module 2×0 to 500 Hz . . . . . 15 - 12

15.6 Counter Module 25/500 kHz . . . . . 15 - 17
15.6.1 Installation Guidelines . . . . . 15 - 20
15.6.2 Data Transfer . . . . . 15 - 25
15.6.3 Functional Description of the Counter Mode . . . . . 15 - 27
15.6.4 Functional Description of the Position Decoder . . . . . 15 - 29
15.6.5 Entering New Setpoints for the Counter and Position Decoder . . . . . 15 - 38
15.6.6 Addressing . . . . . 15 - 39

15.7 Closed-Loop Control Module IP 262 . . . . . 15 - 41

15.8 IP 263 Positioning Module . . . . . 15 - 45

15.9 IP 264 Electronic Cam Controller Module . . . . . 15 - 49

15.10 IP 265 High Speed Sub Control . . . . . 15 - 52

15.11 Positioning Module IP 266 . . . . . 15 - 55

15.12 Stepper Motor Control Module IP 267 . . . . . 15 - 59

15.13 Communications Modules . . . . . 15 - 62
15.13.1 Printer Communications Module CP 521 . . . . . 15 - 62
15.13.2 Communications Module CP 521 BASIC . . . . . 15 - 65

## Appendices


**A Operations List, Machine Code and List of Abbreviations . . . . . A - 1**

A.1 Operations List . . . . . A - 1
A.1.1 Basic Operations . . . . . A - 1
A.1.2 Supplementary Operations . . . . . A - 8
A.1.3 System Operations, for CPU 102 and Higher . . . . . A - 13
A.1.4 Evaluation of CC 1 and CC 0 . . . . . A - 14

A.2 Machine Code Listing . . . . . A - 15

A.3 List of Abbreviations . . . . . A - 18

**B Dimension Drawings . . . . . B - 1**

**C Active and Passive Faults in Automation Equipment . . . . . C - 1**

**D Information for Ordering Accessories . . . . . D - 1**

**E Reference Materials . . . . . E - 1**

**F Siemens Addresses Worldwide . . . . . F - 1**

**Index**

# How to Use This System Manual

The S5-100U is a programmable controller for lower and intermediate performance ranges. It meets all the requirements for a modern programmable controller. To use this controller optimally, you need detailed information.

In this system manual we have attempted to present this information as completely and as well organized as possible. Certain information is repeated in various chapters so that you do not have to leaf through the manual to find what you need.

This How to Use This System Manual section gives you information that will make it easier for you to find what you need. This section explains how the manual is organized.

## Contents of This System Manual

- Hardware Description (Chapters 1, 2, and 3)
  These chapters describe the controllers: how they fit into the SIMATIC® S5 family of programmable controllers, how they function, and how you install them.
- Start-Up Information (Chapters 4, 5, and 6)
  These chapters summarize the information you need to start up your programmable controller. These chapters describe how the hardware and software influence each other.
- The Programming Language of the Programmable Controllers (Chapters 7, 8, and 9)
  These chapters describe the structure, operations, and structuring aids of the STEP® 5 programming language.
- Functions of the Programmable Controllers (Chapters 10, 11, 12, 13)
  Each of these chapters contains a complete description of a particular function, from wiring to programming. Subjects include analog value processing, counter and interrupt inputs, integral clock, and the programmable controller as a SINEC® L1 slave.
- Module Spectrum (Chapters 14 and 15)
  These chapters contain information about all the currently available S5-100U modules that you can use to expand your controller. Chapter 15, Function Modules, includes the modules that require an extensive description (i. e., more than just technical specifications).
- Overviews (Appendices)
  In these chapters you will find not only a complete list of operations but also dimension drawings, a description of errors that may occur during operation of the programmable controller, maintenance and repair procedures, a list of accessories, and reference literature about programmable controllers.

You will find correction pages at the end of the system manual. Use them to indicate any corrections, additions, or suggestions for improvement you might have. Send these suggestions to us. They will help us to improve the next edition of this system manual.

## Conventions

This system manual is organized in menu form to make it easier for you to find information. This means the following:

- Each chapter is marked with printed tabs.
- At the front of the system manual is an overview page that lists the title of each chapter. Following this page, you will find a table of contents.
- At the beginning of each chapter is a table of contents for that chapter. Each chapter has three level headings that are numbered. The fourth level heading is not numbered but appears in **boldface type**.
- Pages, figures, and tables are numbered separately for each chapter. On the back of the table of contents for each chapter you will find a list of the figures and tables that appear in that chapter.

This system manual employs the following specific structuring devices:

- Specific terms have characteristic abbreviations (e. g., programmer is PG). Appendix A contains a list of abbreviations.
- Footnotes are marked with a raised number (e. g., "[1]") or a raised asterisk (" * "). You will find the corresponding explanations in the lower margin of the page or under a figure or table if the footnote appears in one of these.
- Lists are designated with bullets (• as in this particular listing) or with hyphens (-).
- Cross references are indicated as follows: (see section 7.3.2). There are no references to specific page numbers.
- Dimensions in drawings are indicated in millimeters and inches.
- Value ranges are indicated as follows: 17 to 21 or 17-21.
- Especially important information appears in framed boxes such as the following:

**Warning**

You will find definitions for the terms "Warning," "Danger," "Caution," and "Note" in the Safety-Related Guidelines for the User at the end of the introduction.

## Changes Made to the Second Edition of the S5-100U System Manual (Order Number: 6ES5 998-0UB22)

**S5-100U System Manual (Order Number 6ES5 998-0UB23) has been completely revised:**

- The format was adapted to the other system manuals in the SIMATIC S5 family.
- The contents were updated and reorganized.

**Some of the functions of CPU 103 have been expanded:**

- The default settings (default parameters) for DB1 have been integrated into CPU 103 version 8MA03. This feature makes it easier for you to use the internal CPU functions. The following chapters were included or completely revised in the system manual:
  - Chapter 9 “Integrated Blocks and Their Functions”
  - Chapter 12 “Integral Real-Time Clock, for CPU 103 Version 8MA02 and Higher”
  - Chapter 13 “Connecting the S5-100U to SINEC L1, for CPU 102 and Higher”
- The execution times of some operations have been reduced considerably, compared to the “old” CPU 103. For the new execution times refer to the list of operations in Appendix A.

**The S5-100U system has been expanded to include an additional module:**

- The “Communications Module CP 521 BASIC” is described in section 15.10.2.

## Changes Made to the Third Edition of the S5-100U System Manual (Order Number: 6ES5 998-0UB23)

The contents were updated.

## Training

Siemens offers a wide range of training courses for SIMATIC S5 users. Contact your Siemens representative for more information.

## Safety-Related Guidelines for the User

This document provides the information required for the intended use of the particular product. The documentation is written for technically qualified personnel.

Qualified personnel as referred to in the safety guidelines in this document as well as on the product itself are defined as follows.

- System planning and design engineers who are familiar with the safety concepts of automation equipment.
- Operating personnel who have been trained to work with automation equipment and are conversant with the contents of the document in as far as it is connected with the actual operation of the plant.
- Commissioning and service personnel who are trained to repair such automation equipment and who are authorized to energize, de-energize, clear, ground, and tag circuits, equipment, and systems in accordance with established safety practice.

### Danger Notices

The notices and guidelines that follow are intended to ensure personal safety, as well as protect the products and connected equipment against damage.

The safety notices and warnings for protection against loss of life (the users or service personnel) or for protection against damage to property are highlighted in this document by the terms and pictograms defined here. The terms used in this document and marked on the equipment itself have the following significance.

**Danger**

indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

**Warning**

indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

**Caution**

indicates that minor personal injury or property damage can result if proper precautions are not taken.

**Note**

contains important information about the product, its operation or a part of the document to which special attention is drawn.

### Proper Usage

**Warning**

- The equipment/system or the system components may only be used for the applications described in the catalog or the technical description, and only in combination with the equipment, components, and devices of other manufacturers as far as this is recommended or permitted by Siemens.
- The product will function correctly and safely only if it is transported, stored, set up, and installed as intended, and operated and maintained with care.

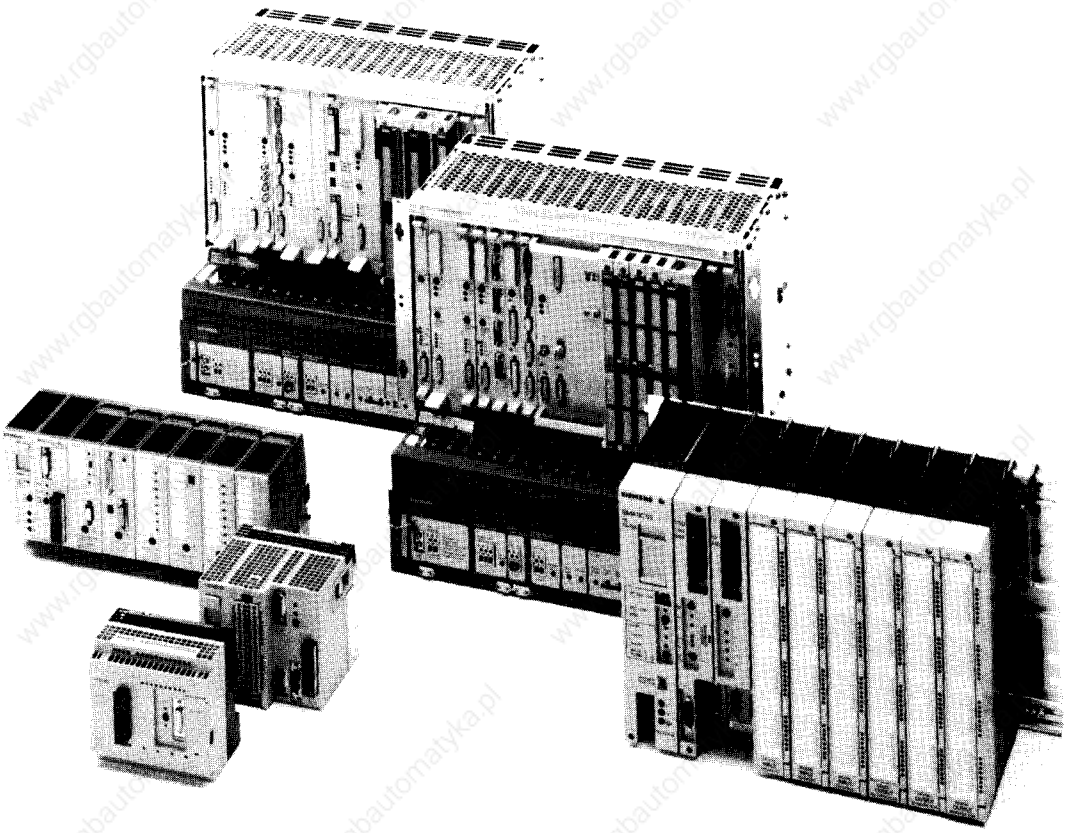
# 1 The SIMATIC S5 System Family

**Figures**

1-1 Members of the SIMATIC S5 System Family . . . . . 1 - 1

# 1 The SIMATIC S5 System Family

The programmable controllers (PLCs) in the SIMATIC S5 family offer economical solutions to simple control tasks and to complex computer functions.



AUT 91 FE 1016

**Figure 1-1. Members of the SIMATIC S5 System Family**

The S5-100U programmable controller is one of the smallest and most economical of the programmable controllers in the SIMATIC S5 family. The S5-100U is especially suited for small automation tasks. It is economical to use these programmable controllers if you want to replace more than five control relays.

The S5-100U has the following features:

- Modular Design

  Depending on the CPU you use, the S5-100U allows you to have a maximum of 448 digital inputs and outputs. It is suitable for machine control and for process automation and monitoring on a medium scale. The S5-100U allows a broad expansion capability with various types of modules to adapt optimally to a control task.

- Rugged, Lightweight Design

  All of the modules you can use with the S5-100U are block-type modules that are small, rugged, and easy to use. The modules operate without fans. None of these modules has electro-magnetically sensitive electronics. The modules are plugged into bus units and screwed tightly so that they are vibration-proof.

  The bus units snap onto a standard mounting rail. You can configure the S5-100U in one or more tiers and configure it vertically or horizontally. The S5-100U offers such a wide range of configuration possibilities that you can use it in rough and difficult operating conditions.

- Simple Programming

  The programming language is STEP 5 and its comprehensive operations set. It provides three different methods of representation, - four, if you have a CPU 103 or higher.

  You can use any of the U series programmers to program your S5-100U, or you can load programs from memory submodules.

## 2 Technical Description


2.1 Programmable Controller Design . . . . . 2 - 1

2.2 Principle of Operation for the Programmable Controller . . . . . 2 - 3
2.2.1 Functional Units . . . . . 2 - 3
2.2.2 Mode of Operation for the External I/O Bus . . . . . 2 - 6

**Figures**


2-1 The S5-100U . . . . . 2 - 1
2-2 Functional Units of the S5-100U . . . . . 2 - 3
2-3 Example of an Arithmetic Logic Unit's Mode of Operation . . . . . 2 - 5
2-4 Accumulator Design . . . . . 2 - 5
2-5 Structure of the External I/O Bus . . . . . 2 - 6
2-6 Data Cycle . . . . . 2 - 7


**Tables**


2-1 Retentive and Non-Retentive Operands . . . . . 2 - 5
2-2 Number of Bits per Module in the Shift Register . . . . . 2 - 8

# 2 Technical Description

This chapter describes the design and principle of operation for the S5-100U programmable controller and its accessories.

## 2.1 Programmable Controller Design

The S5-100U belongs to the SIMATIC S5 range of programmable controllers. The S5-100U consists of various functional units (modules) that you can combine according to the task you want to perform.

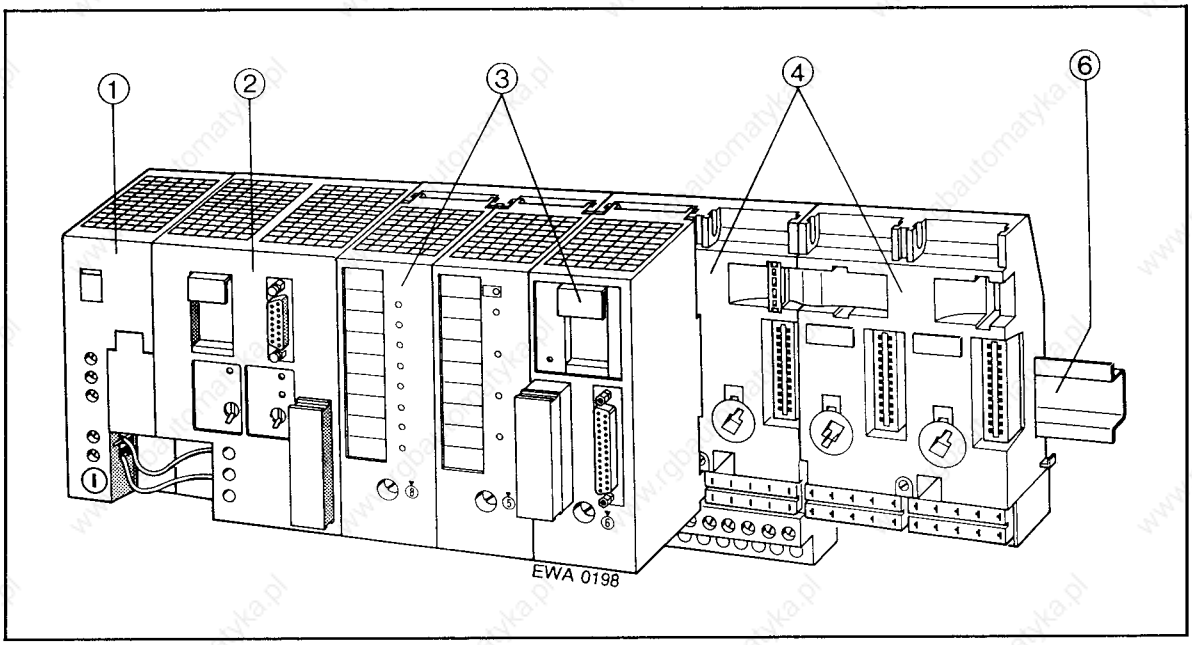



**Figure 2-1. The S5-100U**

① **Power supply module (PS 930)**

This module is required if 24 V DC is not available for the CPU.

② **Central processing unit (CPU)**

The CPU scans the control program. In the event of a power failure, a backup battery located in the battery compartment saves the memory contents (9).
The control program can be stored in a memory submodule.
The CPU has a serial port, and you can connect a programmer, an operator panel, or a SINEC L1 bus to it.

**Input/output modules**

Input/output modules transfer information between the CPU and such process peripherals as sensors, actuators, and transducers. You can use the following types of input/output modules with your S5-100U:

- Digital input modules and digital output modules (4, 8, and 16/16 channel)
  - Use these modules for simple control tasks involving signal states “0” and “1” only.
- Analog input modules and analog output modules
  - Use these modules to record and generate such variable quantities as currents and voltages.
- Timer module
  - Use this module to set various times without having to change the program.
- Counter module
  - Use this module to count pulses up to 500 Hz. You can input comparison values without having to change the program.
- High-speed counter/position detection module
  - Use the high-speed counter to record high-speed counter pulses of 25/500 kHz. You can use this module for position detection in a positioning task.
- Comparator module
  - This module makes it possible for you to monitor preset comparison values, such as for current and voltage.
- Simulator module
  - Use this module to generate digital input signals or to display digital output signals.
- Diagnostic module
  - Use this module to check the function of the I/O bus.
- Communications module (CP)
  - Use this module to output message texts with the date and clock time to a connected printer. You can also use this module to connect to external systems.
- Intelligent I/O module (IP)
  - Use these intelligent input/output modules for such special tasks as temperature control and positioning tasks.

**Bus units with terminal blocks** (Crimp-snap-in or SIGUT, screw type)

Use bus units to connect the CPU to input/output modules. You can plug two input/output modules into a single bus unit.

**Interface modules (IM)**

Use these modules to assemble your S5-100U in a multi-tier configuration.

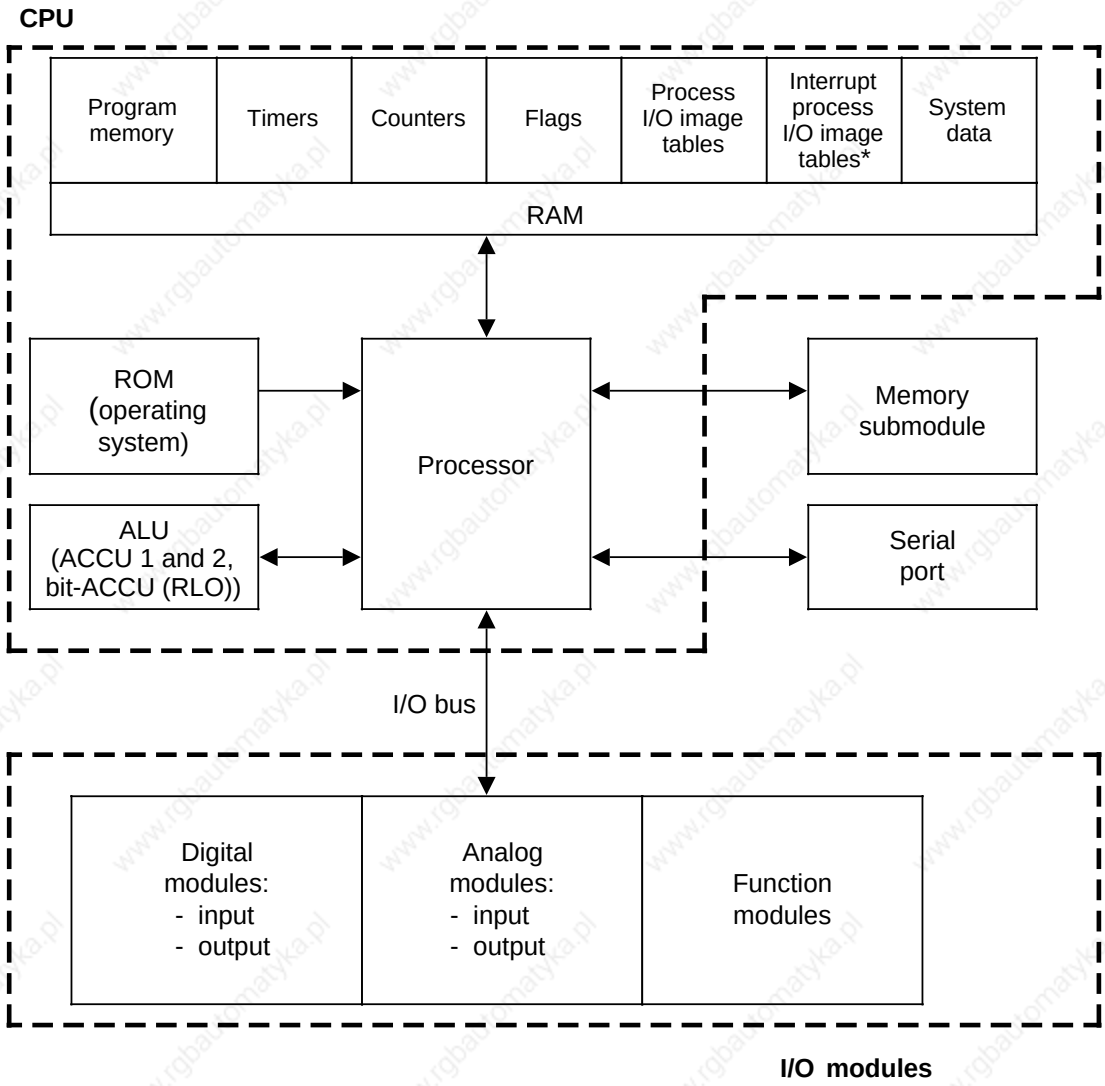
**Standard mounting rail**

Mount your programmable controller on the standard mounting rail.

## 2.2 Principle of Operation for the Programmable Controller

The remainder of this chapter explains how your S5-100U processes your program.

### 2.2.1 Functional Units




* Beginning with CPU 103, version 8MA02

**Figure 2-2. Functional Units of the S5-100U**

### Program Memory (EPROM/EEPROM)

In order to safely store the control program outside of your S5-100U, you must store it on an EPROM or EEPROM memory submodule (see section 4.4).
Programs that are available on a memory submodule (EPROM or EEPROM) can be copied to the internal program memory (see section 4.3). This internal program memory is a reserved area of the CPU's internal RAM memory.

The internal RAM memory has the following characteristics:

- The memory contents can be changed quickly.
- Memory contents are lost when there is a supply voltage failure and there is no battery backup.

### Operating System (ROM)

The operating system contains system programs that determine how the user program is executed, how inputs and outputs are managed, how the memory is divided, and how data is managed.
The operating system is fixed and cannot be changed.

### Process Image Tables (PII, PIQ)

Signal states of input and output modules are stored in the CPU in “process image tables”. Process image tables are reserved areas in the RAM of the CPU.

Input and output modules have the following separate image tables:

- Process image input table (PII)
- Process image output table (PIQ)

### Serial Interface

You can connect programmers, operator panels, and monitors to the serial port (cable connector). You can use the serial port to connect your S5-100U as a slave to the SINEC L1 local area network.

### Timers, Counters, Flags

The CPU has timers, counters, and flags available internally that the control program can use. The program can set, delete, start, and stop the timers and counters. The time and count values are stored in reserved areas of the RAM memory.

There is another area in the RAM memory where information such as intermediate results can be stored as flags. You can address the flags by bits, bytes, or words.

If battery backup is available, then some of the flags and counters remain in the internal RAM memory even if the supply voltage fails or your S5-100U is switched off. These flags and counters are retentive.

Table 2-1 gives information about the number and retentive characteristics (the internal memory contents are retained/are not retained) of these timers, counters, and flags.

**Table 2-1. Retentive and Non-Retentive Operands**

| Operand | Retentive | Non-Retentive | | |
|---|---|---|---|---|
| | CPU 100 to 103 | CPU 100 | CPU 102 | CPU 103 |
| Flags | 0.0 to 63.7 | 64.0 to 127.7 | 64.0 to 127.7 | 64.0 to 255.7 |
| Counters | 0 to 7 | 8 to 15 | 8 to 31 | 8 to 127 |
| Timers | ——— | 0 to 15 | 0 to 31 | 0 to 127 |

### Arithmetic Unit

The arithmetic unit (ALU) consists of two accumulators, ACCU 1 and 2. The accumulators can process byte and word operations.

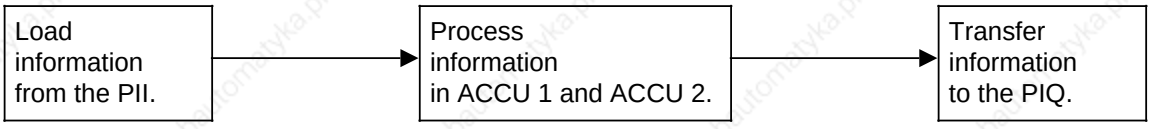



**Figure 2-3. Example of an Arithmetic Logic Unit's Mode of Operation**

Accumulator Design

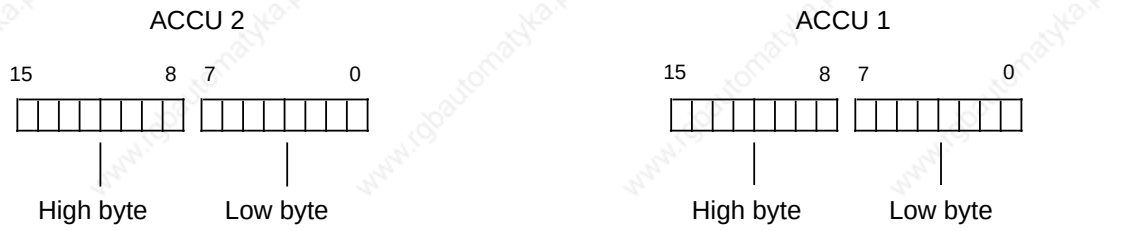



**Figure 2-4. Accumulator Design**

### Processor

According to the control program, the processor calls statements in the program memory in sequence and executes them. It processes the information from the PII and takes into consideration the values of internal timers and counters as well as the signal states of internal flags.

### External I/O Bus

The I/O bus is the electrical connection for all signals that are exchanged between the CPU and the S5-100U modules in a programmable controller.

## 2.2.2 Mode of Operation for the External I/O Bus

The S5-100U has a serial bus for the transfer of data between the CPU and the I/O modules. This serial bus has the following characteristics:

- The modular design permits optimal adaptation to the particular control task.
- No addresses have to be set on the I/O modules.
- A terminating resistor connector is not required.
- Direct access to individual modules is not possible.

A number of shift registers moves the data (Figure 2-5).

Four data bits and one check bit for bus monitoring are assigned to each slot in the bus unit. All modules requiring more than four data bits have their own shift register and therefore do not have to use the shift register of the particular slot.

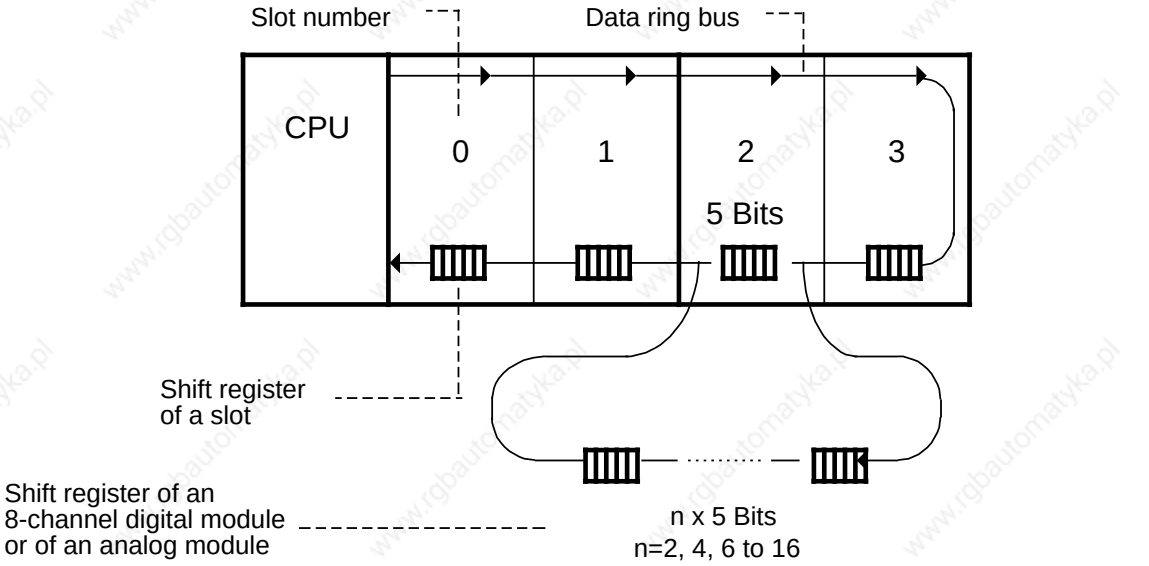



**Figure 2-5. Structure of the External I/O Bus**

**Data Cycle**

Prior to a program scan, the external I/O bus transfers current information from the input modules to the process image input table (PII). At the same time, information contained in the process image output table (PIQ) is transferred to the output modules.

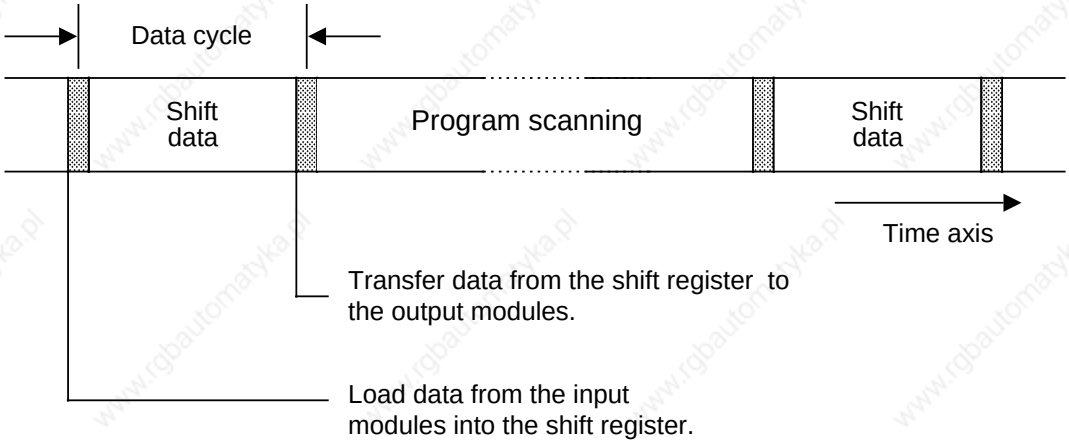



**Figure 2-6. Data Cycle**

**Interrupt Data Cycle**, for CPU 103 version 8MA02 and higher

There is an interrupt input data cycle prior to each time-controlled or interrupt-driven program scan.

Before a time-controlled program scan, current information about the input modules is read into the interrupt PII. Before an interrupt-driven program scan, interrupt inputs on slots 0 and 1 only are read into the interrupt PII.

Following a time-controlled program scan, there is not an interrupt output data cycle until data has been moved into the interrupt PIQ via a transfer operation (see section 6.6.2).
Information is output from the interrupt PIQ to the output modules during an interrupt output data cycle. The PIQ is updated.

### Length of the Shift Register

The total length of the shift register is obtained from the sum of the data bits of all plugged-in modules and of the empty slots. The check bit is not counted.
You must know the length of the shift register to be able to determine the data cycle time. Data cycle time is 25 µs x number of data bits.

**Table 2-2. Number of Bits per Module in the Shift Register**

| Plugged-in Module | Number of Data Bits |
|---|---|
| Diagnostic module or vacant slot | 4 |
| 4-channel digital input and output modules | 4 |
| 500 Hz comparator module, 500 Hz timer module, 500 Hz counter module | 4 |
| 25 KHz counter module | 32 |
| 8-channel digital input and output modules | 8 |
| Digital input and output module, 16 inputs/16 outputs | 16 |
| Simulator module | 8 |
| Analog modules for each activated channel | 16* |
| CP 521, IP 262, IP 266, IP 267 | 64 |
| Refer to the individual manuals for information on other modules. | |

* This does not apply to the 466-8MC11 analog input module (8 data bits).

The CPU specifies the maximum length of the shift register in a particular configuration.

- CPU 100: 256 data bits, 128 (max.) of these from analog modules
- CPU 102: 480 data bits, 256 (max.) of these from analog modules
- CPU 103: 704 data bits, 512 (max.) of these from analog modules

**Note**

If the maximum expansion allowed is exceeded, the S5-100U goes into the STOP mode. The “PEU” bit (I/O not ready) is set in the ISTACK.

**Examples:**

a) CPU 100: This CPU lets you operate six digital modules (8-channel) and two analog modules (4-channel):

[6 x 8+2 x (4 x 16)]=48+128<256

b) CPU 100: This CPU **does not** let you use three digital modules (8-channel) with three analog modules (4-channel) because the maximum permissible number of analog data bits would be exceeded:

[3 x 8+3 x (4 x 16)]=24+192<256

c) CPU 102: This CPU lets you operate seven digital modules (8-channel) and four analog modules (4-channel):

[7 x 8+4 x (4 x 16)]=56+256<480

d) CPU 102: This CPU **does not** let you use 20 digital modules (8-channel) with 5 analog modules (4-channel) because the maximum permissible number of analog data bits would be exceeded:

[20 x 8+5 x (4 x 16)]=160+320=480

e) CPU 103: This CPU lets you operate 24 digital modules (8-channel) and eight analog modules (4-channel):

[24 x 8+8 x (4 x 16)]=192+512=704

f) CPU 103: This CPU **does not** let you use 31 digital modules (8-channel) with four analog modules (2-channel) because the maximum permissible number of slots would be exceeded:

[31 x 8+4 x (2 x 16)]=248+128<704

## 3 Installation Guidelines


| | | |
|---|---|---|
| 3.1 | Installing S5-100U Components | 3 - 1 |
| 3.1.1 | Assembling a Tier | 3 - 1 |
| 3.1.2 | Multi-Tier Expansion | 3 - 5 |
| 3.1.3 | Cabinet Mounting | 3 - 7 |
| 3.1.4 | Vertical Mounting | 3 - 8 |
| 3.2 | Wiring | 3 - 9 |
| 3.2.1 | Connection Methods: Screw-Type Terminals and Crimp Snap-in | 3 - 9 |
| 3.2.2 | Connecting the Power Supply to the S5-100U | 3 - 12 |
| 3.2.3 | Connecting Digital Modules | 3 - 13 |
| 3.2.4 | Connecting the Digital Input/Output Module | 3 - 18 |
| 3.3 | Electrical Configuration | 3 - 20 |
| 3.3.1 | Electrical Configuration for the S5-100U | 3 - 20 |
| 3.3.2 | Electrical Configuration with External I/Os | 3 - 21 |
| 3.3.3 | Non-Floating and Floating Configurations | 3 - 25 |
| 3.4 | Wiring Arrangement, Shielding, and Measures to Guard against Electromagnetic Interference | 3 - 29 |
| 3.4.1 | Running Cables Inside and Outside a Cabinet | 3 - 29 |
| 3.4.2 | Running Cables Outside Buildings | 3 - 30 |
| 3.4.3 | Equipotential Bonding | 3 - 31 |
| 3.4.4 | Shielding Cables | 3 - 32 |
| 3.4.5 | Special Measures for Interference-Free Operation | 3 - 33 |

## Figures


| | | |
|---|---|---|
| 3-1 | Mounting the PS 930 Power Supply Module | 3 - 2 |
| 3-2 | Removing Bus Units | 3 - 3 |
| 3-3 | Coding System to Prevent an Inadvertent Interchange of Modules | 3 - 4 |
| 3-4 | Interconnecting Tiers with Interface Modules (6ES5 316-8MA12) | 3 - 5 |
| 3-5 | Multi-Tier Configuration in a Cabinet with the IM 316 Interface Module (6ES5 316-8MA12) | 3 - 7 |
| 3-6 | Cabinet Mounting with a Series of Devices | 3 - 8 |
| 3-7 | Vertically Mounting a Programmable Controller | 3 - 8 |
| 3-8 | SIGUT/Screw-Type Connection Method | 3 - 9 |
| 3-9 | Mounting the Crimp Snap-in Terminal | 3 - 10 |
| 3-10 | Disconnecting a Terminal | 3 - 11 |
| 3-11 | Connecting a Power Supply Module and a CPU | 3 - 12 |
| 3-12 | Two-Wire Connection of a Sensor to Channel 2 | 3 - 14 |
| 3-13 | Two-Wire Connection of a Lamp to Channel 3 | 3 - 15 |
| 3-14 | Connecting a Sensor to Channel 4 | 3 - 16 |
| 3-15 | Connecting a Lamp to Channel 6 | 3 - 17 |
| 3-16 | Front View of the Digital I/O Module with a Crimp Snap-In Connector (simplified view and not true to scale) | 3 - 18 |
| 3-17 | Connecting a Sensor and a Load to Digital Input/Output Module 482 | 3 - 19 |
| 3-18 | Configuration Possibility: S5-100U with 115/230 V AC Power Supply for Programmable Controller, Sensors, and Actuators | 3 - 22 |
| 3-19 | Configuration Possibility: S5-100U with 24 V DC Power Supply (with Safe Electrical Isolation According to DIN VDE 0160) for Programmable Controller, Sensors, and Actuators | 3 - 23 |
| 3-20 | Non-Grounded Operation; 24 V DC Power Supply (with Safe Electrical Isolation According to DIN VDE 0160) for Programmable Controller and I/Os | 3 - 24 |
| 3-21 | Example: Non-Floating Connection of I/Os to the S5-100U | 3 - 25 |
| 3-22 | Simplified Representation of a Non-Floating I/O Connection | 3 - 26 |
| 3-23 | Simplified Representation of a Galvanically Isolated Connection of the I/Os to the S5-100U | 3 - 27 |
| 3-24 | A Simplified Representation of a Floating I/O Connection | 3 - 28 |
| 3-25 | Laying Equipotential Bonding Conductor and Signal Label | 3 - 31 |
| 3-26 | Fixing Shielded Cables with Various Types of Cable Clamps | 3 - 33 |
| 3-27 | Wiring Coils | 3 - 33 |
| 3-28 | Measures for Suppressing Interference from Fluorescent Lamps in the Cabinet | 3 - 34 |


## Tables


| | | |
|---|---|---|
| 3-1 | Installing, Removing, and Changing S5-100U Components | 3 - 1 |
| 3-2 | Connecting the Load Voltage | 3 - 13 |
| 3-3 | Rules for Common Running of Lines | 3 - 29 |

# 3 Installation Guidelines

## 3.1 Installing S5-100U Components

Except for the I/O module, all of the S5-100U components are mounted on standard mounting rails in accordance with DIN EN 50022-35x15. Mount the rails on a metal plate to obtain the same reference potential.
Bus units with a SIGUT/screw-type, or crimp snap-in connection method have different heights.

If you install, remove, or change any parts of your S5-100U system, your system must be in the state indicated in Table 3-1.

**Table 3-1. Installing, Removing, and Changing S5-100U Components**

| Installing, Removing, and Changing: | S5-100U Power Status | S5-100U Operating Mode | Load Voltage |
|---|---|---|---|
| I/O modules | X | STOP | OFF |
| Bus units<br>Interface modules | Power OFF | X | X |
| CPU power supply | Power supply voltage OFF | X | X |

X=not relevant

### 3.1.1 Assembling a Tier

You need the following components to configure the S5-100U:

- Power supply module
- Central processing unit
- Bus units
- I/O modules

If you do not have a 24 V DC power supply, you must have a power supply module.

Mount the first module on the extreme left end of the standard mounting rail. Add other modules to the right of the first module.

### Mounting the PS 930 Power Supply Module

The backplane design makes it easy to attach this module to the standard mounting rail.

1. Hook the module onto the standard mounting rail.
2. Swing the module back until the slide snaps into place (see Figure 3-1).

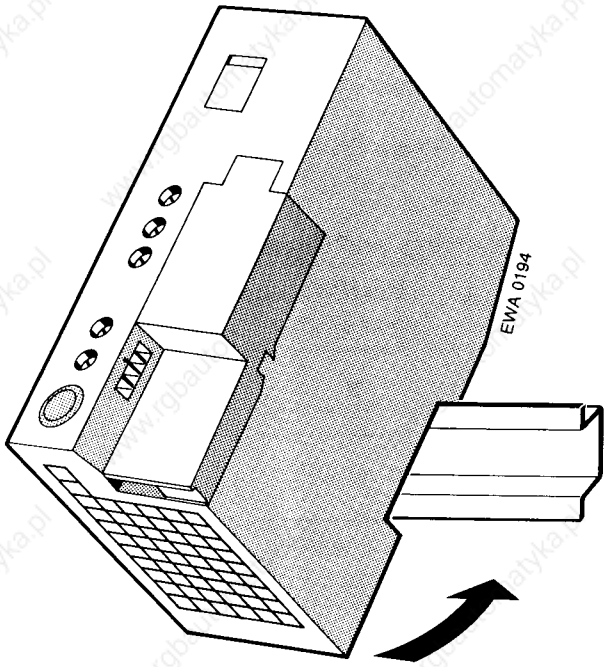



**Figure 3-1. Mounting the PS 930 Power Supply Module**

### Removing the PS 930 Power Supply Module

1. Turn off the 115 V/230 V AC power supply.
2. Remove the connections between the CPU and the power supply module.
3. Use a screwdriver to press down on the slide on the bottom of the module.
4. Swing the module up and out of the standard mounting rail.

### Mounting the Central Processing Unit

Follow the same procedure you used to mount the PS 930 power supply module (see Figure 3-1).

1. Hook the CPU onto the rail and to the right of the power supply module.
2. Swing the CPU back until the slide snaps into place.

### Removing the CPU

1. Remove the I/O module located at slot "0".
2. Pull the connection (ribbon cable) between the CPU and the first bus unit.
3. Pull the connections between the CPU and the power supply module.
4. Use a screwdriver to press down on the slide on the bottom of the module.
5. Swing the module up and out of the standard mounting rail.

### Mounting Bus Units

Use the same procedures to mount the bus unit that you used to mount both the power supply module and the CPU. Hooks are located on the sides of each bus unit. These hooks are used to connect bus units to each other and to connect bus units to the CPU.

### Connecting Bus Units to Each Other or to the CPU

1. Pull the ribbon cable connector located on the top left of the bus unit out of its holder.
2. Plug the connector either into the receptacle located on the right side of the CPU or into the receptacle of the adjacent bus unit located on the left (see Figure 3-2).

### Removing Bus Units

1. Pull the connections to the neighboring bus units or to the CPU.
2. Use a screwdriver to press down on the slide.
3. Swing the module up and out of the standard mounting rail.

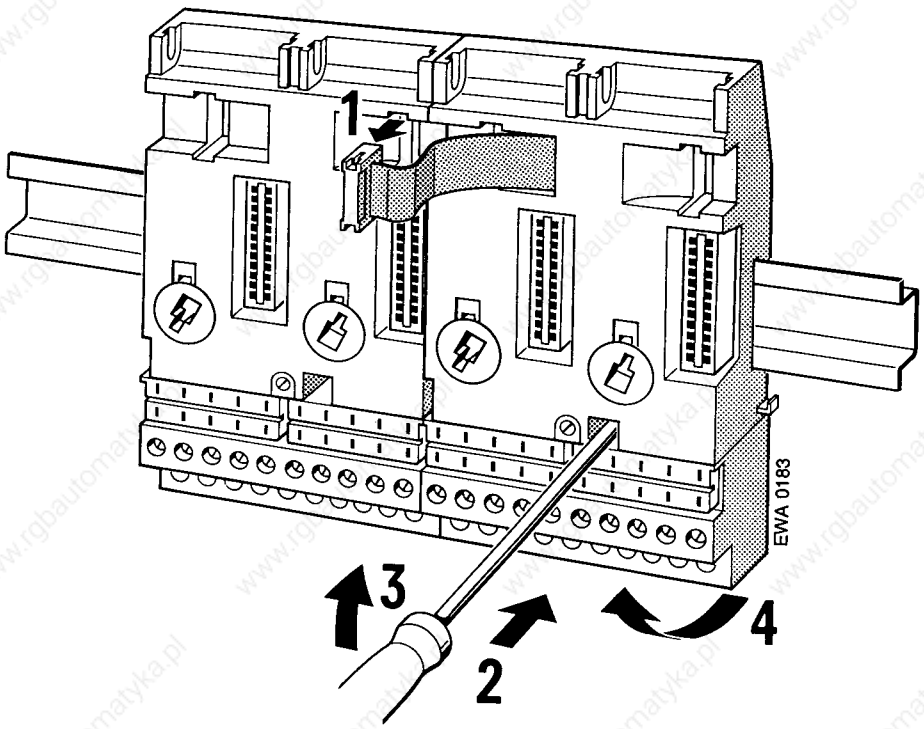



**Figure 3-2. Removing Bus Units**

## Plugging Input and Output Modules into the Bus Units

Before you plug in an input or output module, you must set the bus unit's coding element to match the module type.

### Setting the Coding Element

An identification number is printed on the front plate of every I/O module. Depending on the particular module type, the number is between two and eight. There is a white mechanical coding key located on the back of each module. The position of the coding key is determined by the module type and cannot be changed. The bus unit has a mating component for each key, a white rotating coding element or "lock" (see Figure 3-3).

Use a screwdriver to set the "lock" on the bus unit to the corresponding I/O module code number.

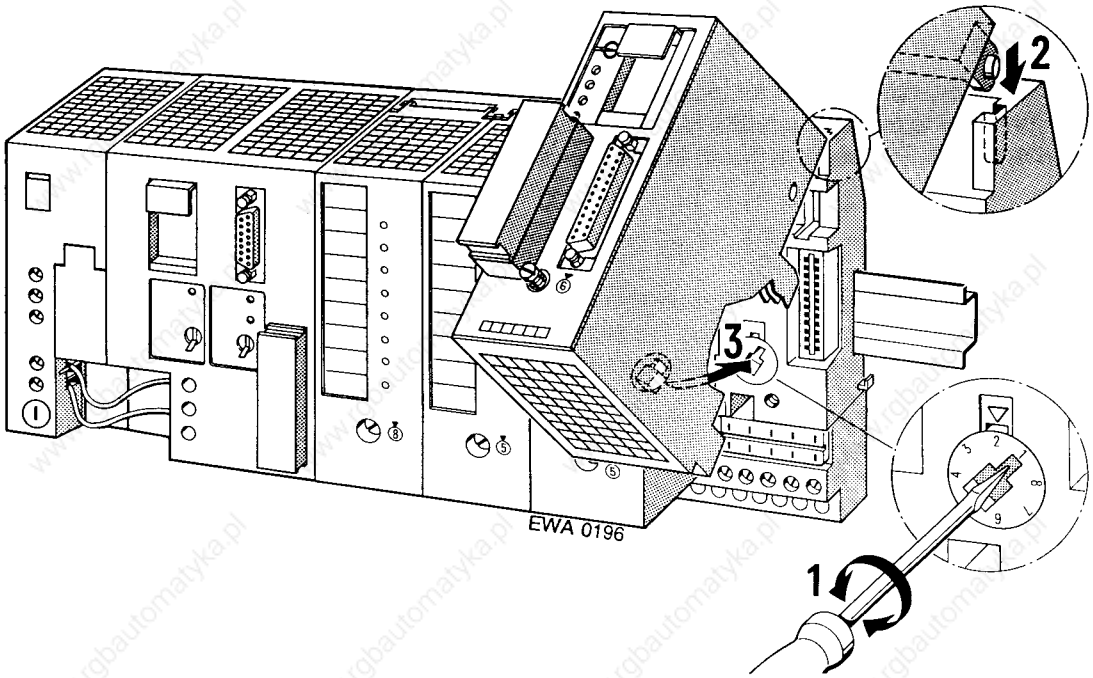



**Figure 3-3. Coding System to Prevent an Inadvertent Interchange of Modules**

The 6ES5 788-8MA11 simulator module does not have a coding key. You can plug in this simulator module in place of any module.

### Attaching I/O Modules

1. Hook the module onto the top of the bus unit.
2. Swing the module down onto the bus unit.
3. Press the module down firmly.
4. Tighten the hold-down screw on the front of the module to attach the module to the bus unit.

### Removing I/O Modules

Remove the hold-down screw and swing the module up and out of the bus unit.

### 3.1.2 Multi-Tier Expansion

If it is not possible to have all of the modules located on one tier, you can expand the configuration up to four tiers. You may use a maximum of 16 bus units. It does not matter how many bus units are mounted on a tier. You need one interface module per tier to interconnect the tiers.

Install an interface module as you would install a bus unit. You must connect each interface module to the last bus unit via the ribbon cable.

Use the IM 315 interface module for two-tier configurations. The IM 315 consists of two modules permanently connected to each other via a 0.5-m (20-in.) cable.

Use the IM 316 interface modules for multi-tier configurations. Use the 712-8 connecting cable to connect the IM 316 interface modules (Order No. 6ES5 712-8...).

The standard mounting rails must have a common reference potential if they are mounted in different cabinets.

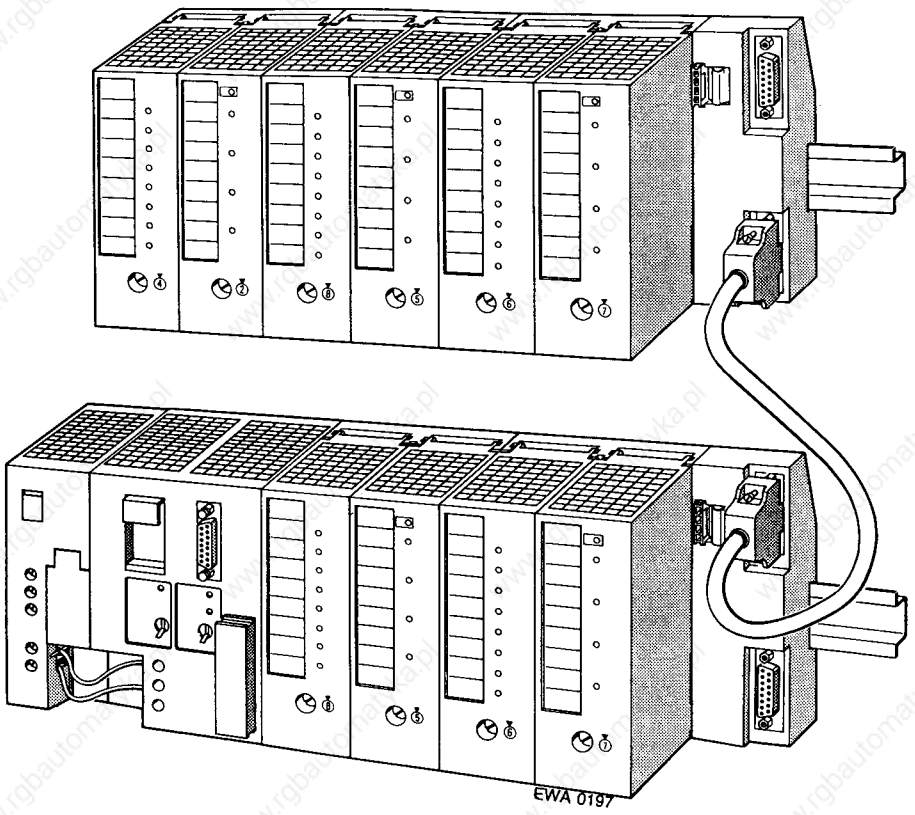



**Figure 3-4. Interconnecting Tiers with Interface Modules (6ES5 316-8MA12)**

**Installing an Interface Module**

1. Hook the interface module to the standard mounting rail.
2. Swing the interface module back until the slide on the bottom snaps into place on the rail.
3. Use the ribbon cable to connect the module to the last bus unit.
4. Use connecting cable 712-8 to join the two interface modules.
5. Connect the cable to the “out” socket on the programmable controller tier and to the “in” socket on the expansion tier.
6. Securely screw the connecting cable plugs in place. Use two screws for each connecting cable plug.

**Removing an Interface Module**

1. Only for the IM 316: Remove the hold-down screws from the plugs and remove the connecting cable.
2. Remove the connecting ribbon cable from the adjacent bus unit.
3. Use a screwdriver to press down on the slide located on the bottom of the interface module.
4. Swing the module up and out of the standard mounting rail.

### 3.1.3 Cabinet Mounting

Make sure that the S5-100U, the power supply, and all modules are well grounded. Mount the S5-100U on a metal plate to help prevent noise. There should be electrical continuity between the grounded enclosure and the mounting rails. Make sure that the system is bonded to earth.

You can use the 8LW system or the 8LX system mounting plates (see Catalog NV 21).

Adequate ventilation and heat dissipation are important to the proper operation of the system. You must have at least 210 mm (8.3 in.) between each mounting rail (see Figures in Appendix B) for proper ventilation.

Always locate the power supply and the CPU on the lowest tier to ensure better heat dissipation. To measure cabinet ventilation, define the total heat loss by calculating the sum of all typical heat losses (see Catalog ST 52.1).

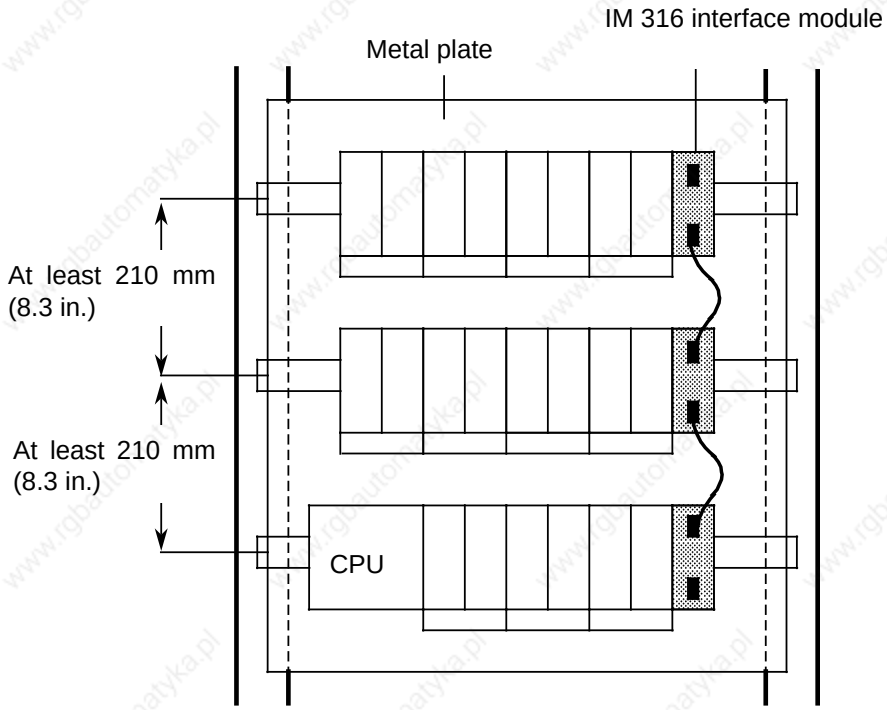



**Figure 3-5. Multi-Tier Configuration in a Cabinet with the IM 316 Interface Module (6ES5 316-8MA12)**

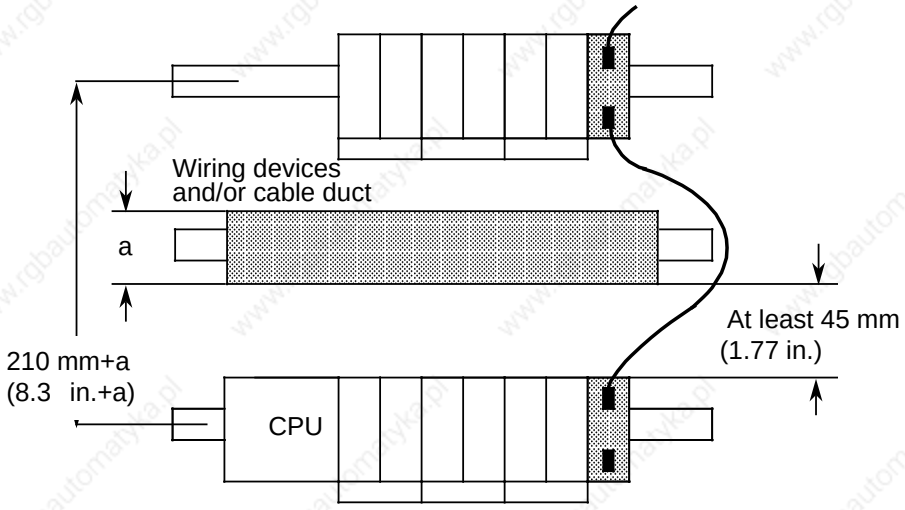



**Figure 3-6. Cabinet Mounting with a Series of Devices**

### 3.1.4 Vertical Mounting

You can also mount the standard mounting rails vertically and then attach the modules one over the other. Because heat dissipation by convection is less effective in this case, the maximum ambient temperature allowed is 40 °C (104 °F).

Use the same minimum clearances for a vertical configuration as for a horizontal configuration.

You must install a clamp (see Catalog SA 2) on the lower end of the programmable controller tier to hold the modules mechanically in position.

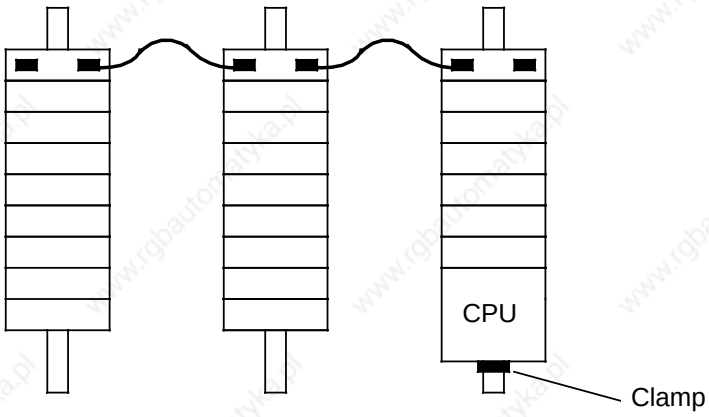



**Figure 3-7. Vertically Mounting a Programmable Controller**

## 3.2 Wiring

### 3.2.1 Connection Methods: Screw-Type Terminals and Crimp Snap-in

**SIGUT Screw-Type Terminal**

When using screw-type terminals, you can clamp two cables per terminal. It is best to use a 3.5-mm screwdriver to tighten the screws.

Permissible cable cross-sections are:

- A stranded conductor with a core end sleeve: 2 x 0.5 to 1.5 $mm^2$
- A solid conductor: 2 x 0.5 to 2.5 $mm^2$

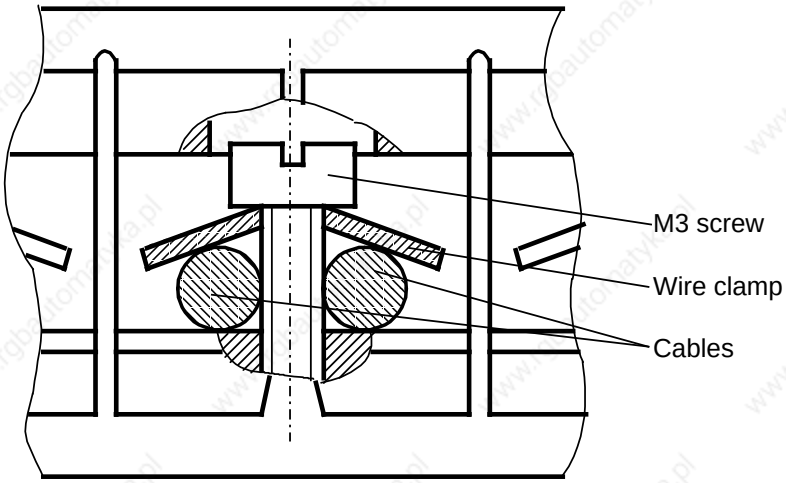



**Figure 3-8. SIGUT/Screw-Type Connection Method**

### Crimp Snap-in Terminals

Bus units using the crimp snap-in connection method have the same height as the CPU.

You can connect stranded conductors with a cross-section of 0.5 to 1.5-mm$^2$ to these terminals.

### Connecting the Contact to the Terminal Block

Refer to Figure 3-9 and perform the following steps to connect the contact to the terminal block.

1. Remove the module that is plugged into the bus unit.
2. Use a screwdriver to press down on the terminal block (1).
3. Swing the terminal block up. The rear side is now visible (2).
4. Push the contact into the desired opening until the locating spring engages.
   - Caution: The spring must point into the slot.
5. Pull lightly on the cable to make certain that the contact is properly engaged.
6. Swing the terminal block back into its original position.
7. Press up on the terminal block until it snaps into position.

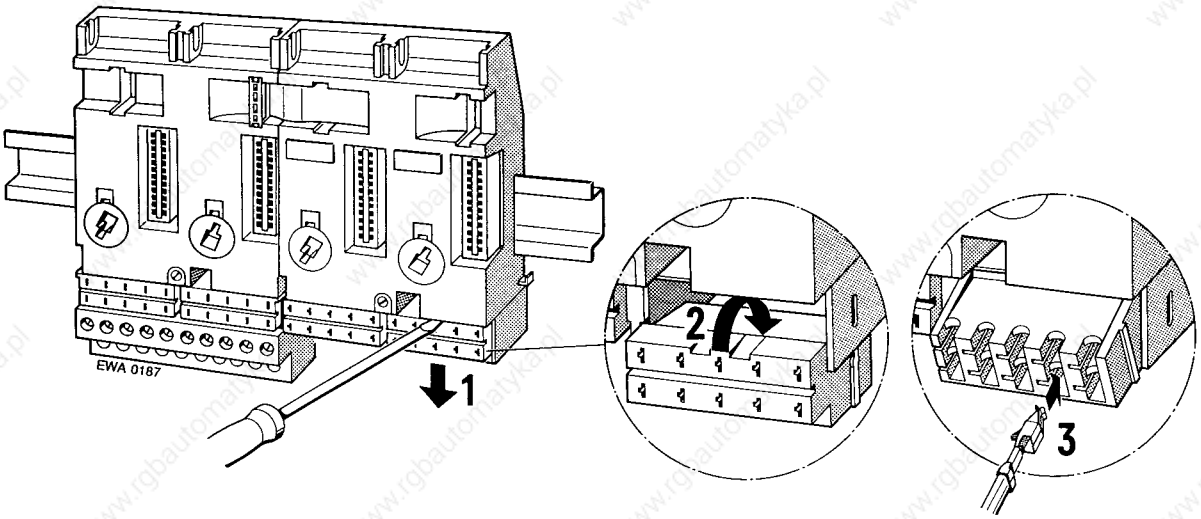



**Figure 3-9. Mounting the Crimp Snap-In Terminal**

### Disconnecting a Terminal

1. Position the terminal block as is shown in Figure 3-10.
2. Insert the extraction tool into the slot beside the terminal so that you can compress the barb.
3. Position the cable in the groove on the extraction tool and pull out both the tool and the cable.
4. Realign the deformed barb so that you can use the terminal again.

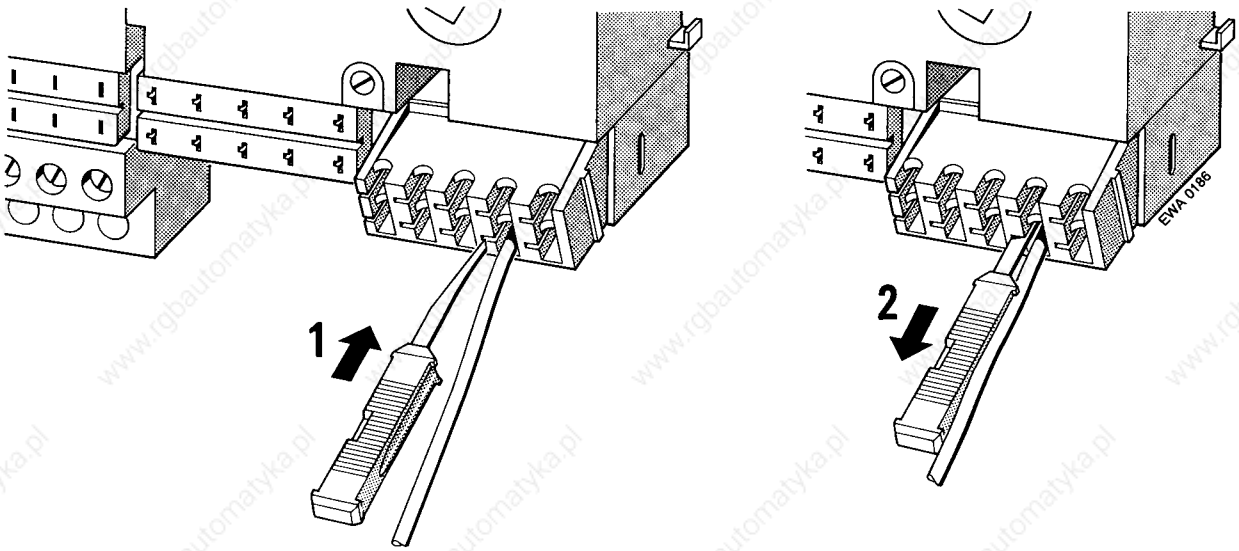



**Figure 3-10. Disconnecting a Terminal**

## 3.2.2 Connecting the Power Supply to the S5-100U

### Power Supply Module

1. Set the voltage selector to the supply voltage you are using.
2. Swing up the protective cover.
3. Connect the supply cable to terminals L1, N and ⏚ (see Figure 3-11).
4. Close the protective cover.

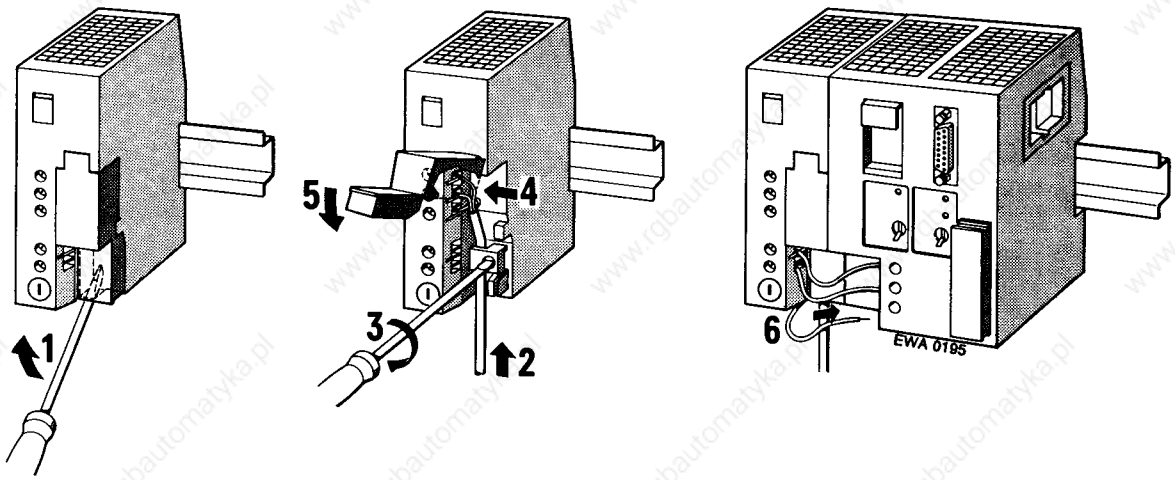



**Figure 3-11. Connecting a Power Supply Module and a CPU**

115/230 V AC power supply modules can be operated with a load voltage of 120/230 V AC.

### CPU

1. Connect the L+ and M terminals of the PS 931 power supply module to the corresponding terminals on the CPU (see Figure 3-23).
2. Connect the ⏚ terminal of the CPU to the standard mounting rail.

### 3.2.3 Connecting Digital Modules

All I/O modules are plugged into bus units. Connect the I/O modules to the terminal blocks of the bus units. The connections illustrated in this section are of the screw terminal type (SIGUT connection method).

You can also use the crimp snap-in connection method described in section 3.2.1. In both cases, the terminal assignments are marked on the terminal blocks.

The assignments listed in Table 3-2 always apply for connecting the load voltage.

**Table 3-2. Connecting the Load Voltage**

| Load Voltage | Terminal 1 | Terminal 2 |
|---|---|---|
| 24 V DC | L+ | M |
| 115/230 V AC | L1 | N |

* 115/230 V AC digital modules can be operated with a load voltage of 120/230 V AC.

**Note**

For digital outputs, energy is temporarily stored in an internal capacitor for about 100 ms after the L+ supply is switched off.
Please note that this energy may be sufficient to activate low-rating loads (e.g., pulse valves) for a triggered output.

### Connecting Four-Channel Digital Modules

All of these modules are designed for a two-wire connection. You can therefore wire directly to the sensor or output field device. An external distribution block is not required.

The four channels of a module are numbered from .0 through .3. (Numbers .4 through .7 are only significant for the ET 100 distributed I/O system.) Each channel has a pair of terminals on the terminal block.

The terminal assignments and the connection diagram are printed on the front plate of the module.

### Connecting Four-Channel Input Modules

**Example:** Connecting a sensor to channel 2 (address I 3.2) on the input module in slot 3 (see Figure 3-12)

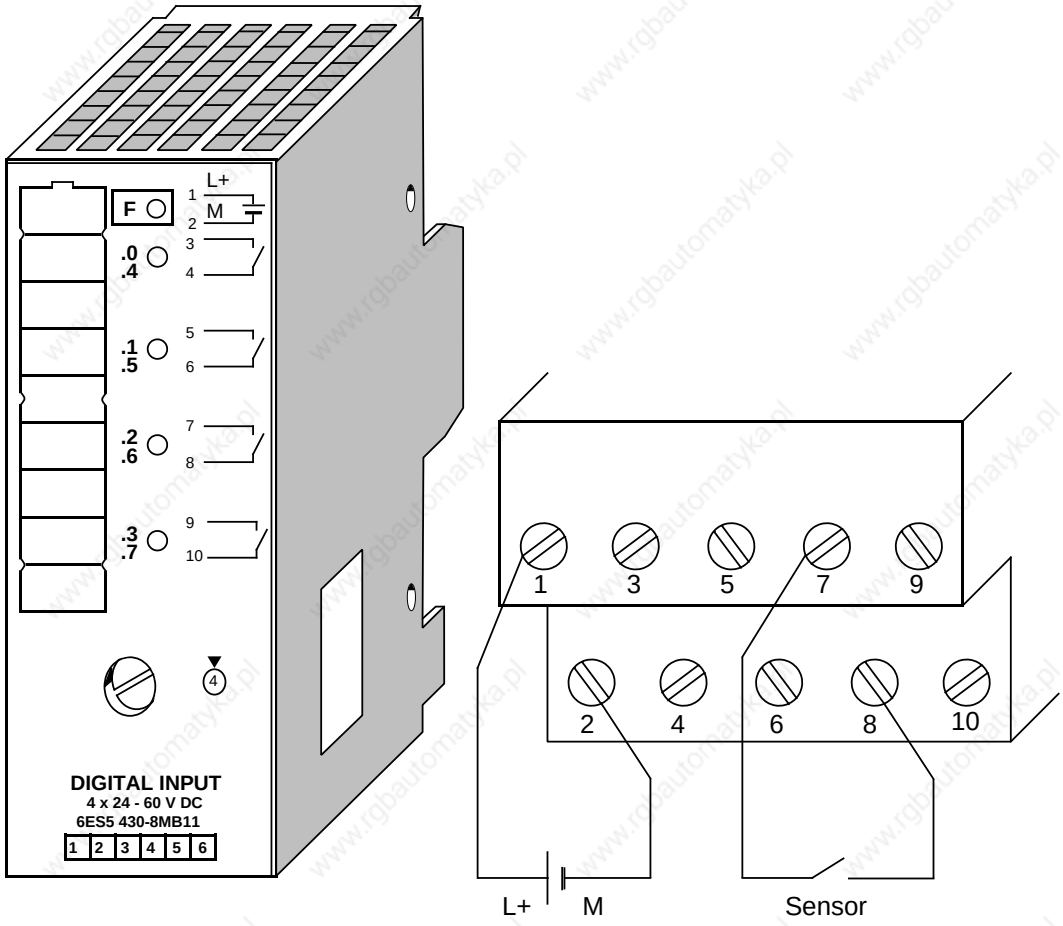



**Figure 3-12. Two-Wire Connection of a Sensor to Channel 2**

### Connecting Four-Channel Output Modules

**Example**: Connecting a lamp to channel 3 (address Q 1.3) on the output module in slot 1 (see Figure 3-13)

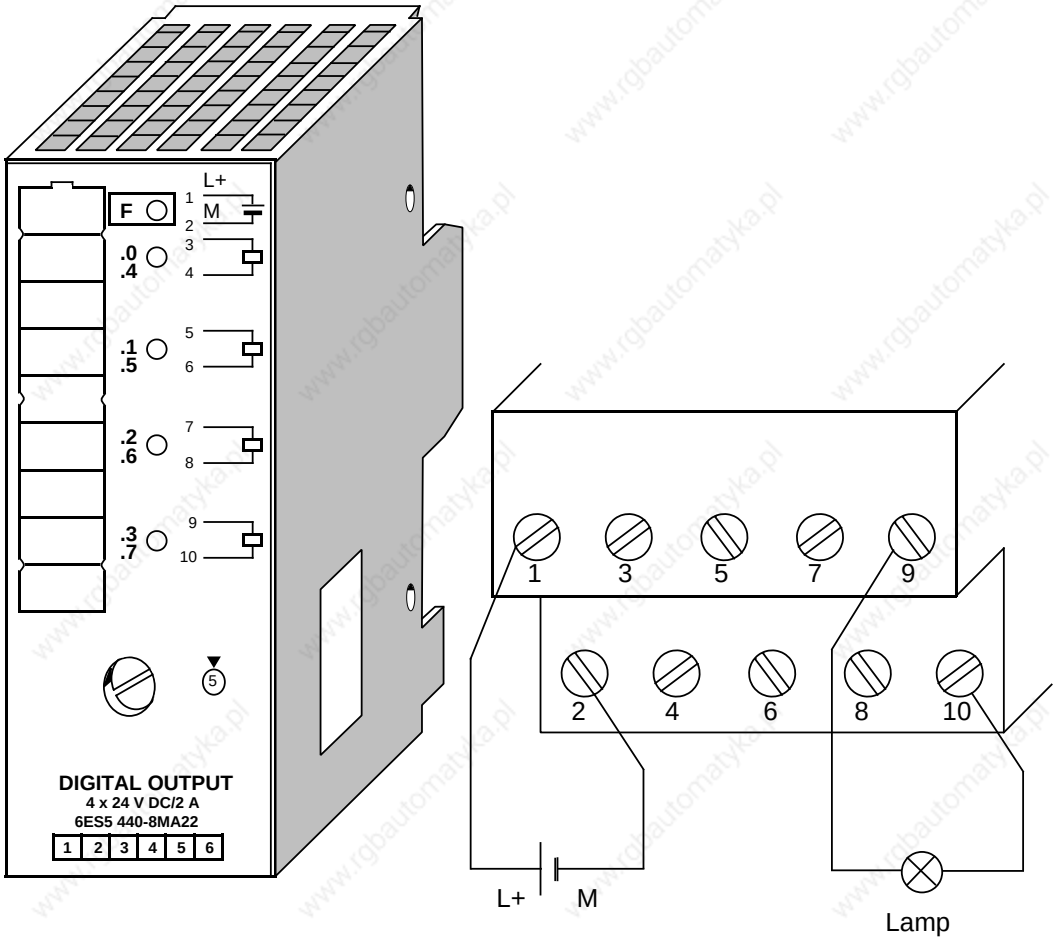



**Figure 3-13. Two-Wire Connection of a Lamp to Channel 3**

### Connecting Eight-Channel Digital Modules

These modules do not have a two-wire connection. You therefore need an external distribution block.

The eight channels of a module are numbered from .0 through .7. One terminal on the terminal block is assigned to each channel. The terminal assignment and the connection diagram are printed on the front plate of the module.

### Connecting Eight-Channel Input Modules

The sensors must be connected to terminal 1 via the L+ terminal block.

**Example:** Connecting a sensor to channel 4 (address I 3.4) on an input module in slot 3 (see Figure 3-14)

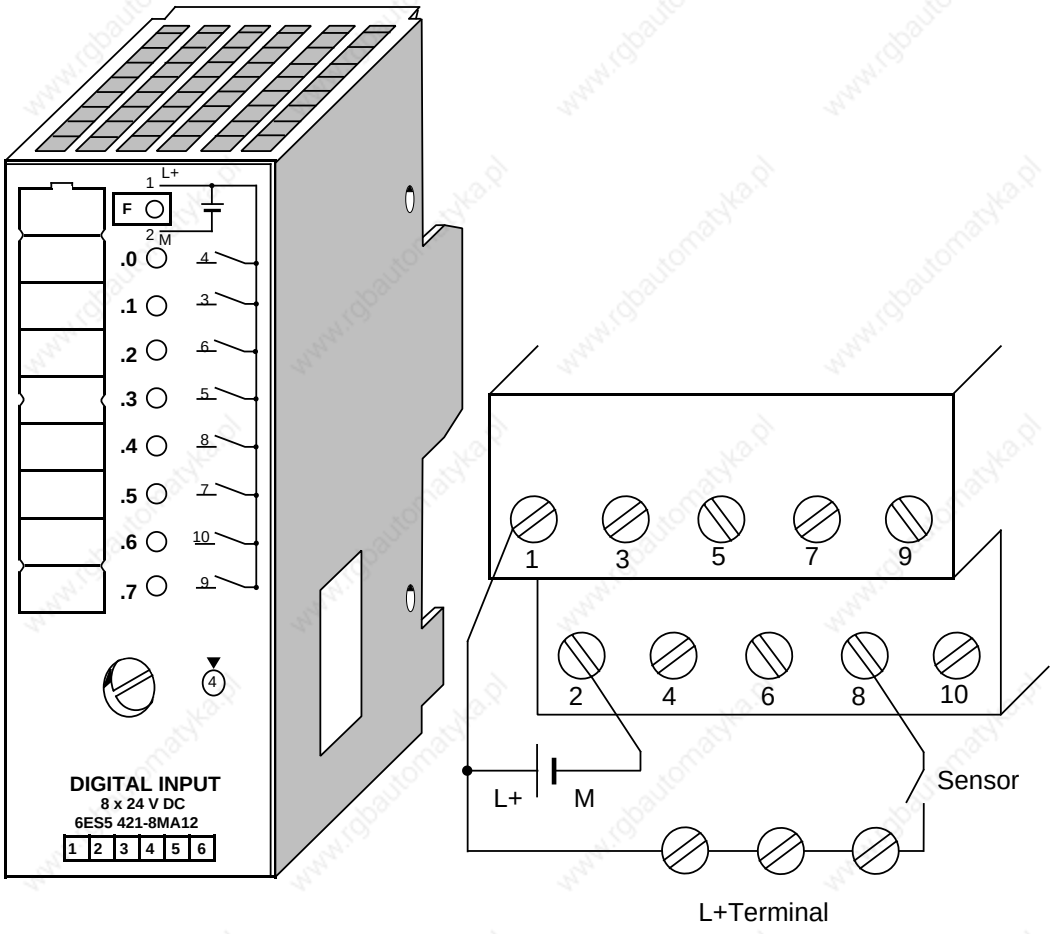



**Figure 3-14. Connecting a Sensor to Channel 4**

### Connecting Eight-Channel Output Modules

The actuators must be connected to terminal 2 via the M (negative) terminal block. This does not apply to the digital output module 8× 5 to 24 V DC/0.1 A (see section 14.6.2).

**Example:** Connecting a lamp to channel 6 (address output Q 5.6) on an output module in slot 5 (see Figure 3-15)

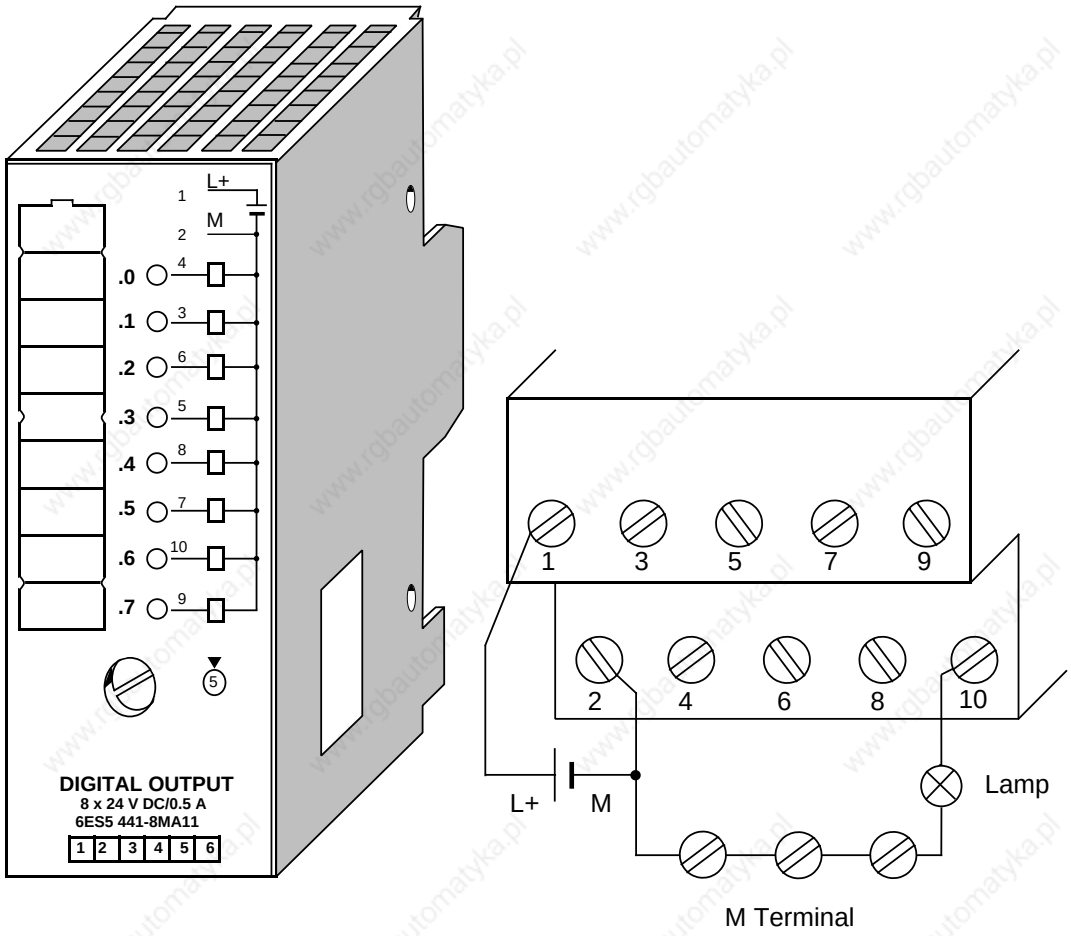



**Figure 3-15. Connecting a Lamp to Channel 6**

### 3.2.4 Connecting the Digital Input/Output Module

Use only slots 0 through 7 when you plug the module into the bus unit. Use a 40-pin cable connector with a screw-type connection or crimp snap-in connection for wiring. The module does not have a two-wire connection. You must therefore use an external distribution block.

Every channel is assigned a terminal on the 40-pin connector. The channel numbers are printed on the front plate.

The 16 channels on the input side (IN) are numbered from n.0 through n.7 and from n+1.0 through n+1.7. The 16 channels on the output side (OUT) are numbered from n.0 through n.7 and from n+1.0 through n+1.7. "n" is the start address of the slot. Slot 0, for example, has the start address of n=64 (see chapter 6).

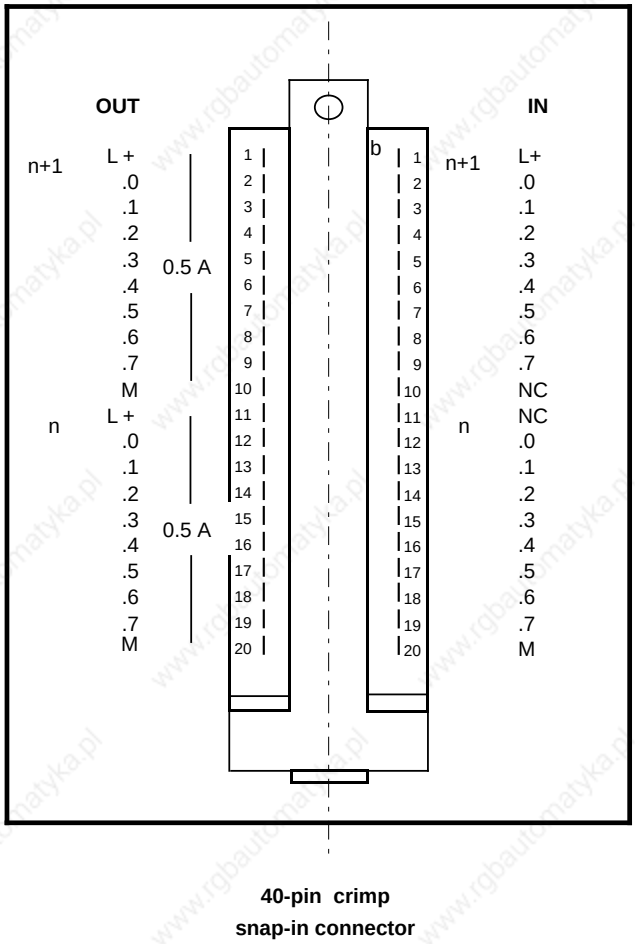



**Figure 3-16. Front View of the Digital I/O Module with a Crimp Snap-In Connector (simplified view and not true to scale)**

**Example:** The start address for the modules is 65.3. Inputs and outputs have the same address. A sensor is to be connected to input I 64.4 and a lamp to output Q 7.3. Figure 3-17 illustrates the wiring on the front connector.

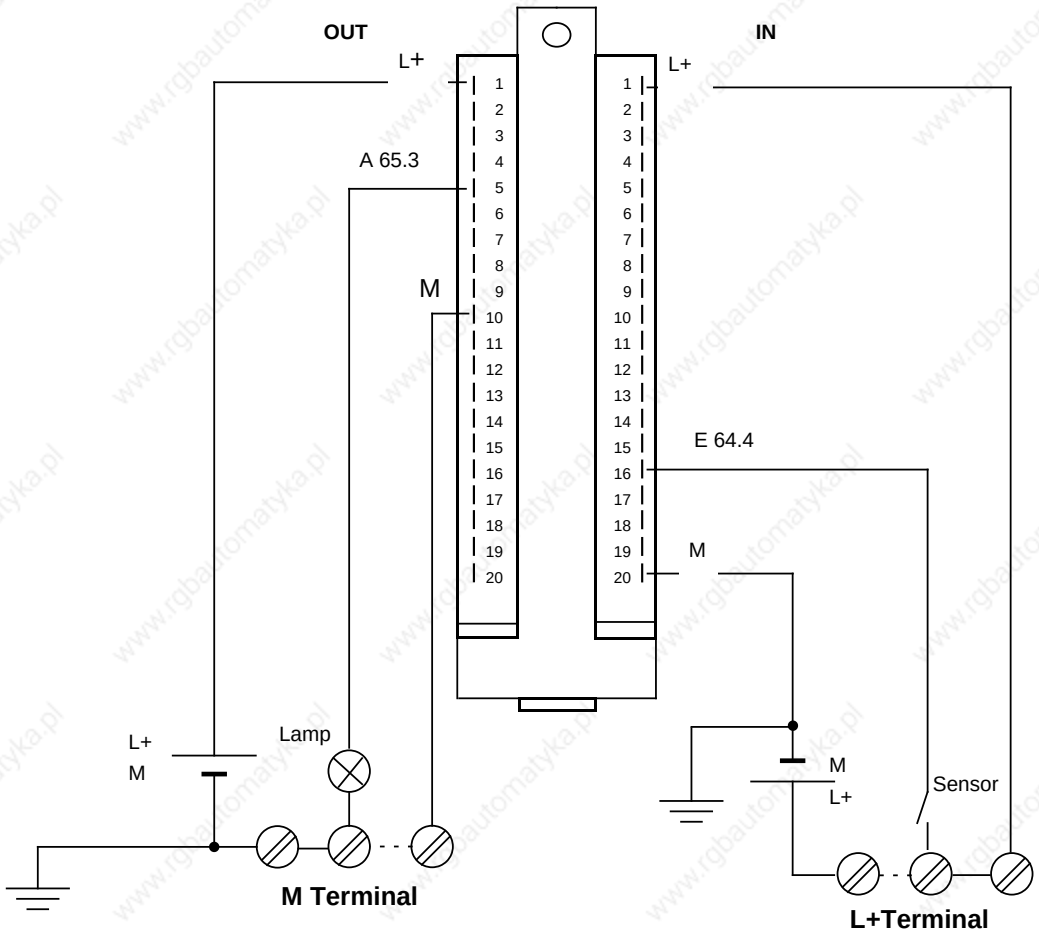



**Figure 3-17. Connecting a Sensor and a Load to Digital Input/Output Module 482**

> **Note**
>
> Chapter 11 describes how to connect analog modules.

## 3.3 Electrical Configuration

### 3.3.1 Electrical Configuration for the S5-100U

**Power Supply**

The entire control for the S5-100U consists of the following separate electrical circuits:

- Control circuit for the S5-100U (24 V DC)
- Control circuit for the sensors (24 V DC)
- Load circuit for the actuators (24 V DC or 115/230 V AC)

**Control Circuit**

The power source for the control circuit supplies the CPU, the bus units, the programmer interface, and the internal control circuits for the I/O modules. When the incoming supply is 24 V DC/1 A, the PS 931 power supply module provides an internal supply of +9 V up to a total of 1 A current input to the I/O modules. The grounding spring on the CPU forces the control circuit to be connected to the standard mounting rail. The grounding spring must also be protected from interference. The grounding spring must be grounded.

**Load Circuit**

The power source for the load circuit supplies the actuators of the process peripherals.

It is suggested that you use one of the following for a 24 V DC power supply:

- The PS 931 power supply module (see Chapter 14)
- A Siemens load power supply from the 6EV1 series (see Appendix D)

If you use load power supplies other than the recommended ones, make certain that the load voltage is in the range of 20 to 30 V (including ripple).

**Note**

If you use a switched-mode power supply unit to supply floating analog modules and BEROs, then this supply must be filtered through a network.

You can connect several mutually independent load circuits adjacent to each other on a single programmable controller. These connections can either be non-floating or floating (see section 3.3.3).

### 3.3.2 Electrical Configuration with External I/Os

Figures 3-18, 3-19, and 3-20 display different configuration possibilities. Pay attention to the following points when you design your configuration. The numbers appearing in parentheses in the following points refer to the numbers in Figures 3-18 to 3-20.

- You must have a main switch (**1**) in accordance with VDE 0100 for your S5-100U, the sensors, and the actuators.
- You do not need an additional fuse (**2**) to connect your S5-100U and the load circuit to power if your radial lines are a maximum of 3 meters (9.84 feet) long and are inherently earth-fault proof and short-circuit proof.
- You need a load power supply (**3**) for 24 V DC load circuits.
  - You need a back-up capacitor (rating: 200 µF per 1 A of load current) if you have non-stabilized load power supplies.
- If you have AC load circuits, galvanic isolation via a transformer (**4**) is recommended.
- You should ground the load circuit at one end. Provide a removable connection (**5**) to the ground conductor on the load power supply (terminal M) or on the isolating transformer.
  - You must provide earth-fault monitoring for any non-grounded load circuits.
- You must separately fuse (**6** and **7**) the load voltage for sensor circuits and for actuator circuits.
- You must connect the standard mounting rail of the S5-100U to the ground conductor through a capacitor (**8**, to suppress high-frequency noise) for a non-grounded configuration.
- You must have a low-resistance connection between the standard mounting rail and the cabinet's chassis ground (**10**) for a grounded configuration.
- You need a power fuse (**9**) to protect against a short-circuit occurring in the power supply.

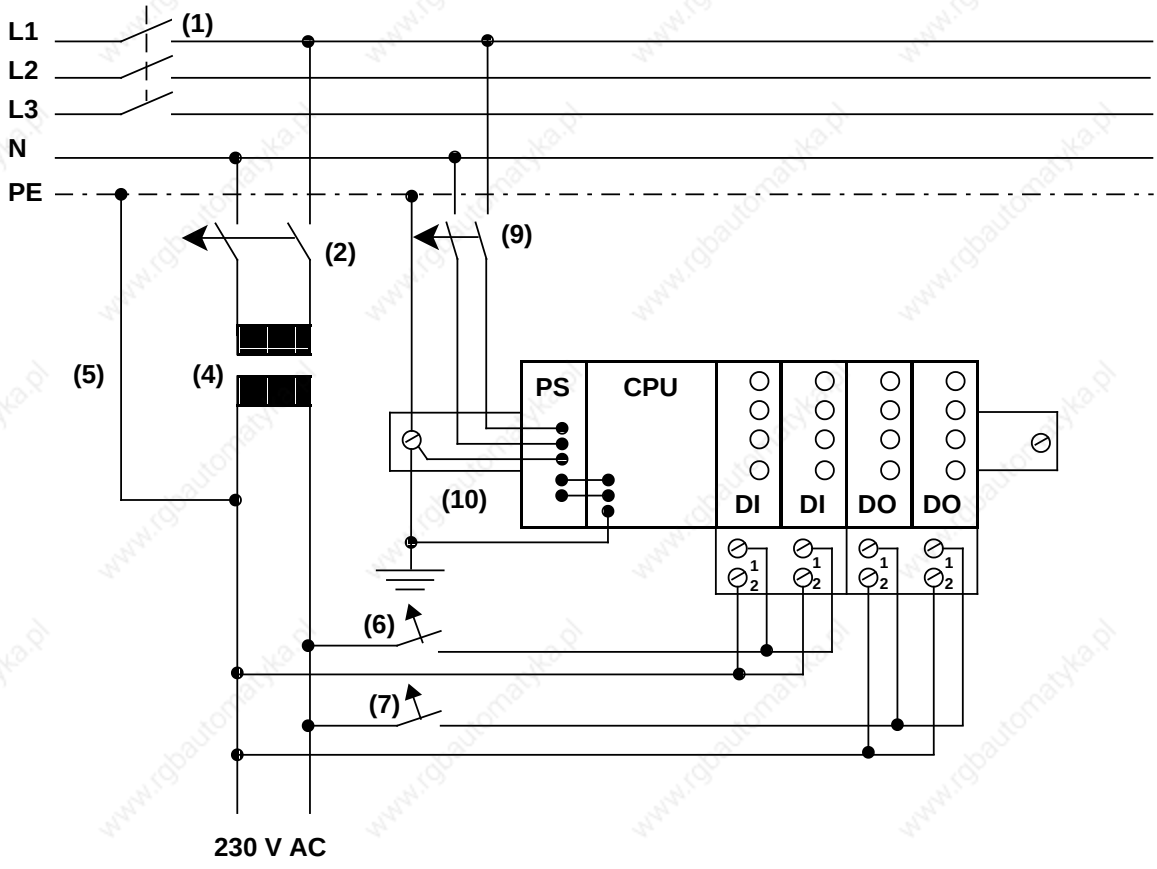



**Figure 3-18. Configuration Possibility: S5-100U with 115/230 V AC Power Supply for Programmable Controller, Sensors, and Actuators**

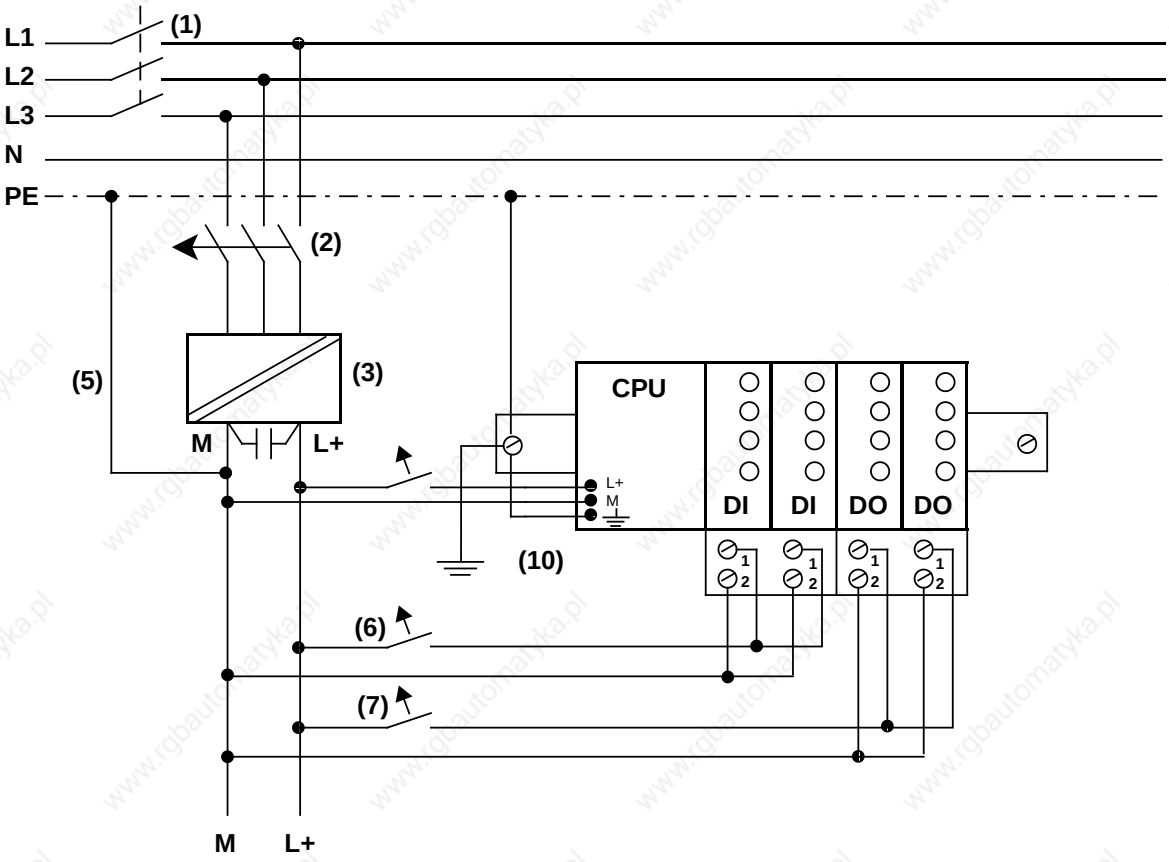



**Figure 3.19 Configuration Possibility: S5-100U with 24 V DC Power Supply (with Safe Electrical Isolation According to DIN VDE 0160) for Programmable Controller, Sensors, and Actuators**

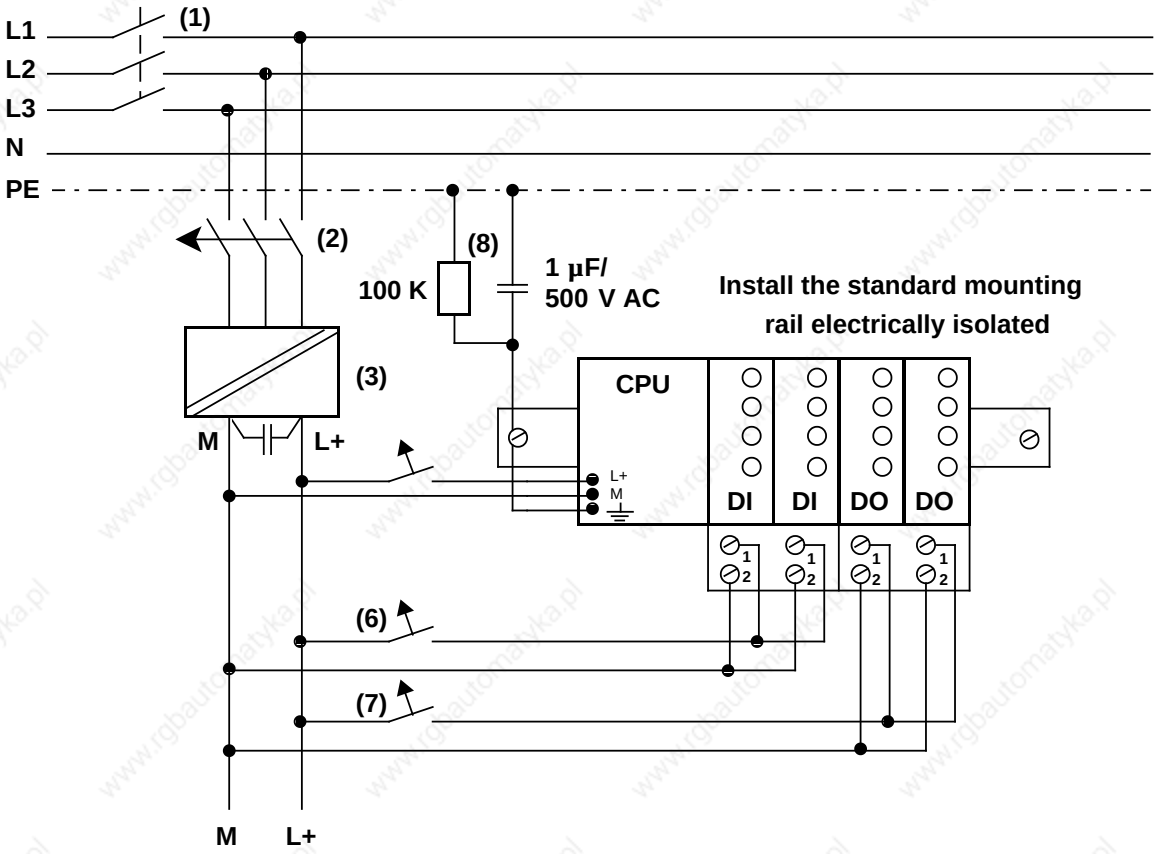



**Figure 3-20. Non-Grounded Operation; 24 V DC Power Supply (with Safe Electrical Isolation According to DIN VDE 0160) for Programmable Controller and I/Os**

Interference voltages are discharged to the ground conductor (PE) via a capacitor. You can prevent static charges by connecting a high-ohmic resistor (approx. 100 k / W) parallel to the capacitor.

### 3.3.3 Non-Floating and Floating Configurations

The S5-100U is powered by its own control circuit. The I/Os are powered by the load circuit.

The circuits can either be connected to the same grounding point (non-floating) or galvanically isolated (floating).

**Example of a Non-Floating Connection of Digital Modules**

A 24 V DC load circuit has the same chassis grounding as the control circuit of the CPU.

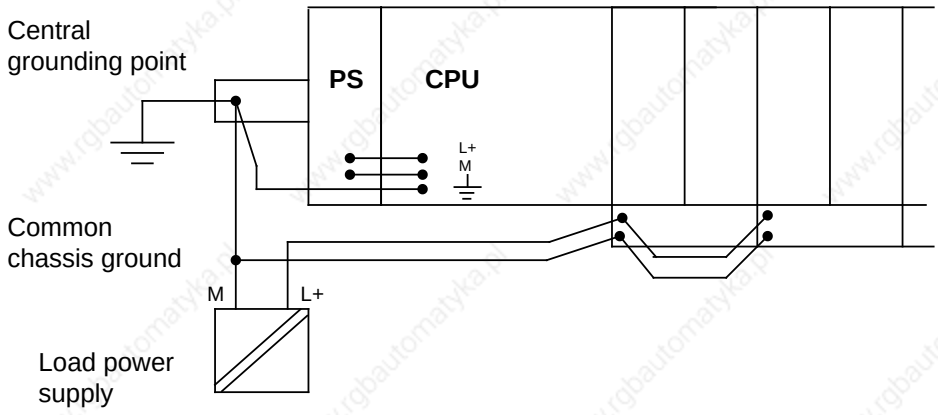



**Figure 3-21. Example: Non-Floating Connection of I/Os to the S5-100U**

The common chassis grounding connection makes it possible for you to use reasonably priced non-floating I/Os. These modules function according to the following principles.

- Input modules
  - The ground line, line M (control circuit chassis) is the reference potential. A voltage drop $V_1$ on line affects the input signal level $V_I$.
- Output modules
  - Terminal 2 (M) of the terminal block is the reference potential. A voltage drop $V_2$ on the line raises the chassis potential of the output driver and thus reduces the resulting control voltage $V_{CV}$.

Figure 3-22 shows a simplified connection of the S5-100U with a non-floating external I/O.

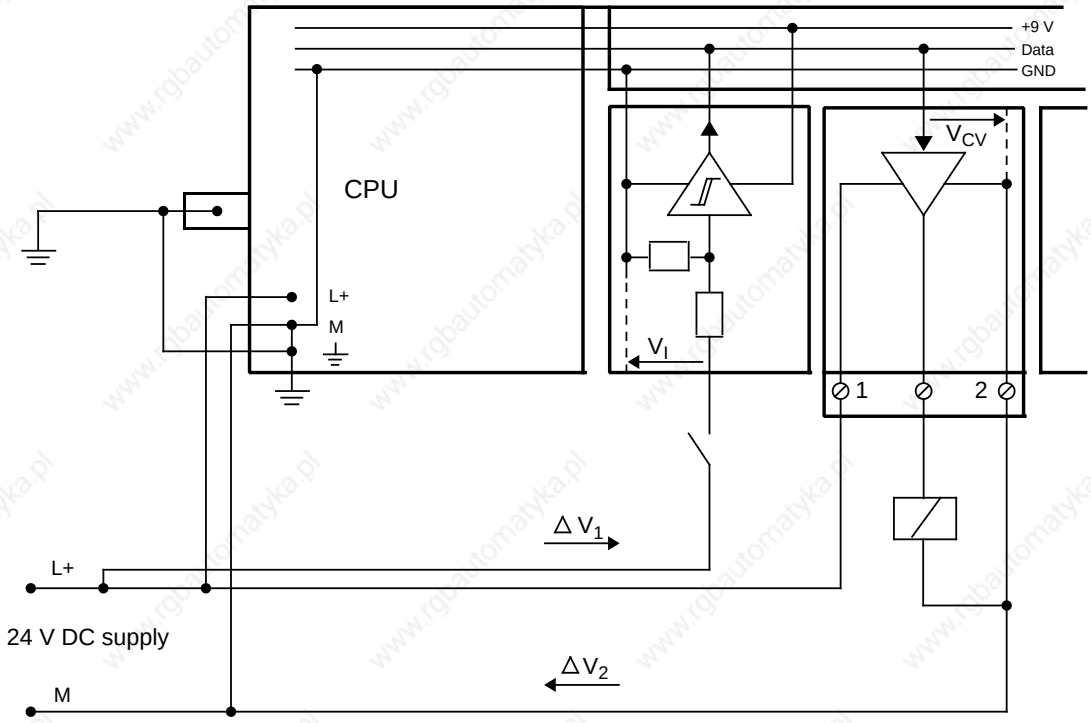



**Figure 3-22. Simplified Representation of a Non-Floating I/O Connection**

When you have a non-floating configuration, you must make certain that the voltage drop on cables and does not exceed 1 V. If 1 V is exceeded, the reference potentials could change and the modules could malfunction.

**Warning**

If you use non-floating I/O modules, you must provide an external connection between the chassis ground of the non-floating I/O module and the chassis ground of the CPU.

**Example of a Floating Configuration with Digital Modules**

Floating configuration is required in the following situations.

- When you need to increase interference immunity in the load circuits
- When load circuits cannot be interconnected
- When you have AC load circuits

If you have a floating configuration, the PLC's control circuit and the load circuit must be galvanically isolated.

Figure 3-23 shows a simplified connection of galvanically isolated I/Os.

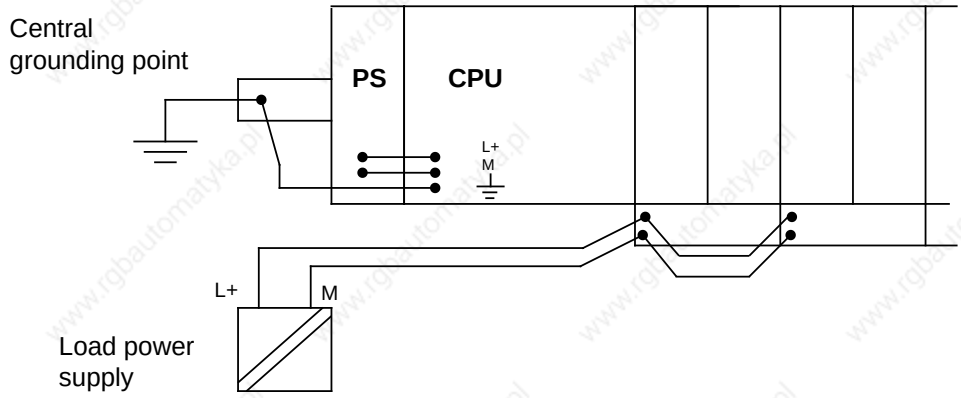



**Figure 3-23. Simplified Representation of a Galvanically Isolated Connection of the I/Os to the S5-100U**

Figure 3-24 shows a simplified schematic for the connection of floating I/O modules.

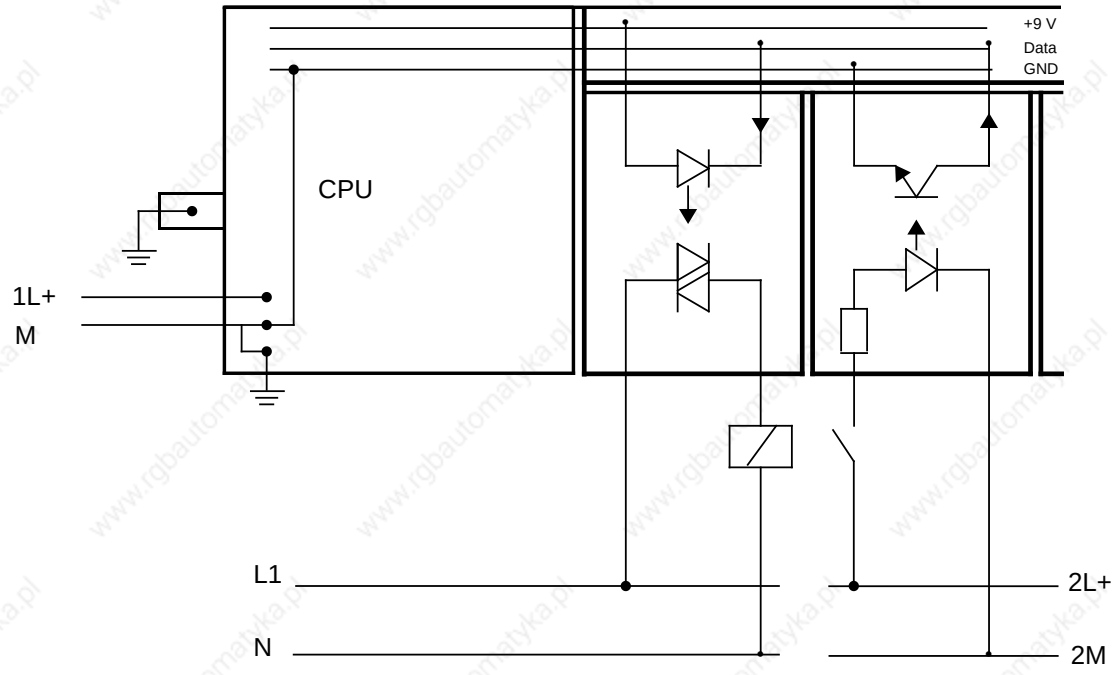



**Figure 3-24. A Simplified Representation of a Floating I/O Connection**

## 3.4 Wiring Arrangement, Shielding and Measures against Electromagnetic Interference

This section describes the wiring arrangements for bus cables, signal cables, and power supply cables that guarantee the **e**lectro**m**agnetic **c**ompatibility (EMC) of your installation.

### 3.4.1 Running Cables Inside and Outside a Cabinet

Dividing the lines into the following groups and running the groups separately will help you to achieve electromagnetic compatibility (EMC).

Group A: Shielded bus and data lines (for programmer, OP, printer, SINEC L1, Profibus, Industrial Ethernet, etc.)
Shielded analog lines
Unshielded lines for DC voltage 60 V
Unshielded lines for AC voltage 25 V
Coaxial lines for monitors

Group B: Unshielded lines for DC voltage > 60 V and 400 V
Unshielded lines for AC voltage > 25 V and 400 V

Group C: Unshielded lines for AC voltage > 400 V

You can use the following table to see the conditions which apply to the running of the various combinations of line groups.

**Table 3-3. Rules for Common Running of Lines**

| | Group A | Group B | Group C |
|---|---|---|---|
| **Group A** | | | |
| **Group B** | | | |
| **Group C** | | | |

Legend for table:

Lines can be run in common bundles or cable ducts
Lines must be run in separate bundles or cable ducts (without minimum distance)
Inside cabinets, lines must be run in separate bundles or cable ducts and outside cabinets but inside buildings, lines must be run on separate cable trays with a gap of a least of 10 cm between lines.

### 3.4.2 Running Cables Outside Buildings

Run lines outside buildings where possible in metal cable supports. Connect the abutting surfaces of the cable supports galvanically with each other and ground the cable supports.

When you run cables outdoors, you must observe the regulations governing lightning protection and grounding. Note the general guidelines:

**Lightning Protection**

If cables and lines for SIMATIC S5 devices are to be run outside buildings, you must take measures to ensure internal and external lightning protection.

Outside buildings run your cables
either

- In metal conduits grounded at both ends
  or
- In steel-reinforced concrete cable channels

Protect signal lines from overvoltage by using:

- Varistors
  or
- Lightning arresters filled with inert gas

Install these protective elements at the point where the cable enters the building.

> **Note**
>
> Lightning protection measures always require an individual assessment of the entire system. If you have any questions, please consult your local Siemens office or any company specializing in lightning protection.

**Grounding**

Make certain that you have sufficient equipotential bonding between the devices.

### 3.4.3 Equipotential Bonding

Potential differences may occur between separate sections of the system if

- Programmable controllers and I/Os are connected via non-floating interface modules or
- Cables are shielded at both ends but grounded via different sections of the system.

Potential differences may be caused, for instance, by differences in the system input voltage. These differences must be reduced by means of equipotential bonding conductors to ensure proper functioning of the electronic components installed.

Note the following for equipotential bonding:

- A low impedance of the equipotential bonding conductor makes equipotential bonding more efficient.
- If any shielded signal cables connected to earth/protective earth at both ends are laid between the system sections concerned, the impedance of the additional equipotential bonding conductor must not exceed 10 % of the shield impedance.
- The cross-section of the equipotential bonding conductor must be matched to the maximum compensating currents. The following cross-sections are recommendable:
  - 16 mm² copper wire for equipotential bonding line up to 200 m (656.2 ft).
  - 25 mm² copper wire for equipotential bonding line over 200 m (656.2 ft).
- Use equipotential bonding conductors made of copper or zinc-plated steel. Equipotential bonding conductors are to be connected to earth/protective earth via a large contact area and to be protected against corrosion.
- The equipotential bonding conductor should be laid in such a way as to achieve a relatively small contact area between equipotential bonding conductor and signal cables (see Figure 3-25).

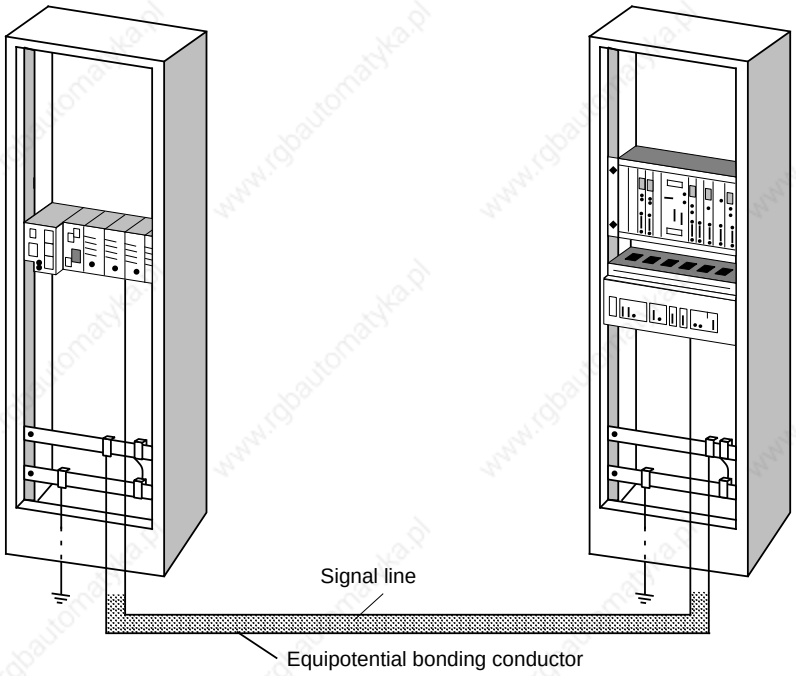



**Figure 3-25. Laying Equipotential Bonding Conductor and Signal Cable**

### 3.4.4 Shielding Cables

Shielding is a measure to weaken (attenuate) magnetic, electric or electromagnetic interference fields.

Interference currents on cable shields are discharged to ground over the shield bar which has a conductive connection to the housing. So that these interference currents do not become a source of noise in themselves, a low-resistance connection to the protective conductor is of special importance.

Use only cables with shield braiding if possible. The effectiveness of the shield should be more than 80%. Avoid cables with foil shielding since the foil can easily be damaged by tension and pressure; this leads to a reduction in the shielding effect.

As a rule, you should always shield cables at both ends. Only shielding at both ends provides good suppression in the high frequency range.

As an exception only, you can connect the shielding at one end. However, this attenuates only the lower frequencies. Shielding at one end can be of advantage in the following cases:

- If you cannot run an equipotential bonding conductor
- If you are transmitting analog signals (e.g. a few microvolts or microamps)
- If you are using foil shields (static shields).

Always use metallic or metalized connectors for data lines for serial connections. Secure the shield of the data line at the connector housing. Do ***not*** connect the shield to the PIN1 of the connector strip!
In the case of stationary operation, you are recommended to insulate the shielded cable without interrupt and to connect it to the shield/protective ground bar.

**Note**

If there are potential differences between the earthing points, a compensating current can flow over the shielding that is connected at both ends. For this reason, connect an additional equipotential bonding conductor.

Note the following when connecting the cable shield:

- Use metal cable clamps for fixing the braided shield. The clamps have to enclose the shield over a large area and make good contact (see Figure 3-26).
- Connect the shield to a shield bar immediately at the point where the cable enters the cabinet. Route the shield to the module; do **not** connect it to the module.

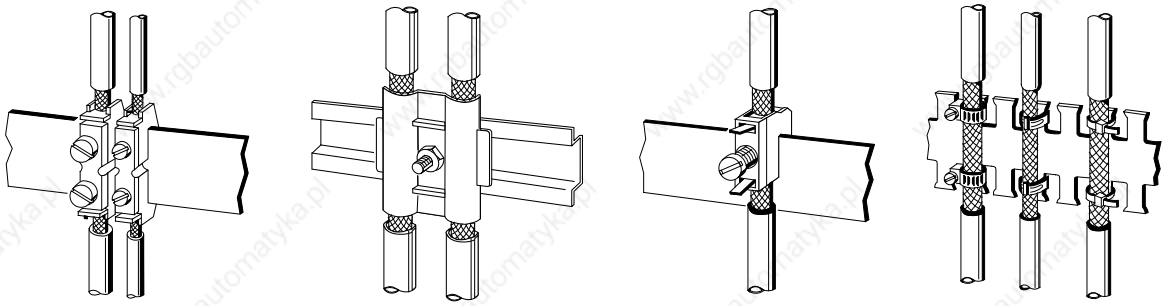


**Figure 3-26. Fixing Shielded Cables with Various Types of Cable Clamps**

## 3.4.5 Special Measures for Interference-Free Operation

**Arc Suppression Elements For Inductive Circuits**

Normally, inductive circuits (e.g. contactor or relay coils) energized by SIMATIC S5 do not require to be provided with external arc suppressing elements since the necessary suppressing elements are already integrated on the modules.

It only becomes necessary to provide arc supressing elements for inductive circuits in the following cases:

- If SIMATIC S5 output circuits can be switched off by additionaly inserted contactors (e.g. relay contactors for EMERGENCY OFF). In such a case, the integral suppressing elements on the modules become ineffective.
- If the inductive circuits are ***not*** energized by SIMATIC S5.

You can use free-wheeling diodes, varistors or RC elements for wiring inductive circuits.

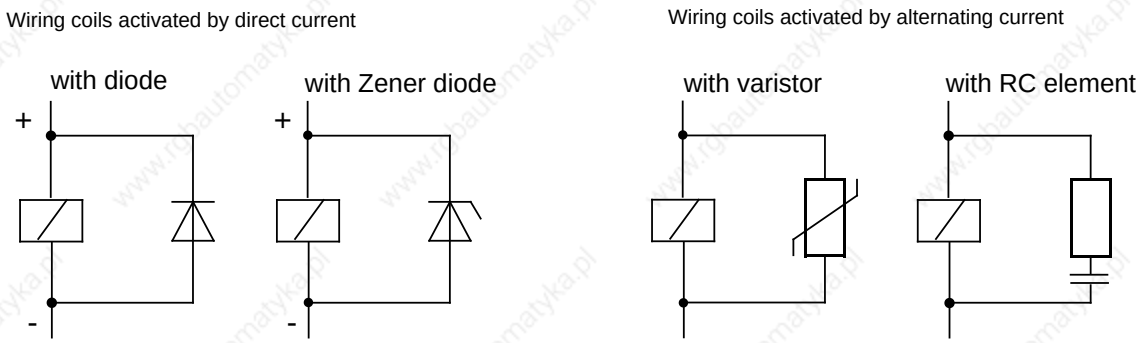



**Figure 3-27. Wiring Coils**

### Mains Connection for Programmers

Provide a power connection for a programmer in each cabinet. The plug must be supplied from the distribution line to which the protective ground for the cabinet is connected.

### Cabinet Lighting

Use, for example, LINESTRA® lamps for cabinet lighting. Avoid the use of fluorescent lamps since these generate interference fields. If you cannot do without fluorescent lamps, you must take the measures shown in Figure 3.28.

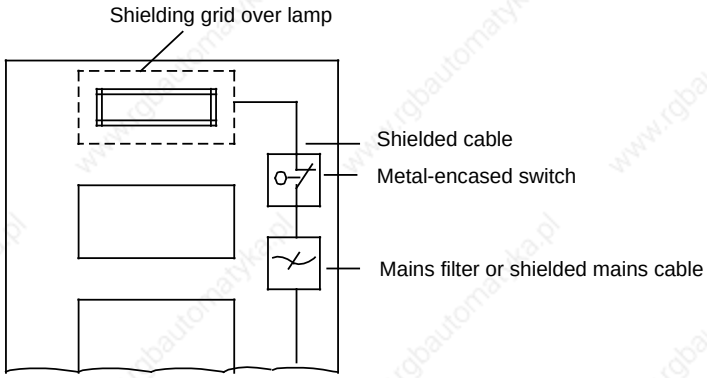



**Figure 3-28. Measures for Suppressing Interference from Fluorescent Lamps in the Cabinet**

## 4 Start-up and Program Tests


| | | |
|---|---|---|
| 4.1 | Operating Instructions | 4-1 |
| 4.1.1 | CPU Operator Panel | 4-1 |
| 4.1.2 | Operating Modes | 4-1 |
| 4.1.3 | Performing an Overall Reset on the Programmable Controller | 4-2 |
| 4.2 | Starting Up a System | 4-3 |
| 4.2.1 | Suggestions for Configuring and Installing the Product | 4-3 |
| 4.2.2 | Procedures for Starting Up the Programmable Controller | 4-4 |
| 4.3 | Loading the Program into the Programmable Controller | 4-5 |
| 4.4 | Backing Up the Program | 4-7 |
| 4.4.1 | Backing Up the Program on a Memory Submodule | 4-7 |
| 4.4.2 | Function of the Back-Up Battery | 4-8 |
| 4.5 | Program-Dependent Signal Status Display “STATUS” | 4-8 |
| 4.6 | Direct Signal Status Display “STATUS VAR” | 4-9 |
| 4.7 | Forcing Outputs, “FORCE”, for CPU 103 and Higher | 4-10 |
| 4.8 | Forcing Variables, “FORCE VAR” | 4-10 |
| 4.9 | Search Function | 4-11 |
| 4.10 | Program Check, for CPU 103 and Higher | 4-11 |

**Figures**


| | | |
|---|---|---|
| 4-1 | CPU Operator Panel | 4 - 1 |
| 4-2 | Procedure for Loading the Program Automatically | 4 - 5 |
| 4-3 | Procedure for Loading the Program Manually | 4 - 6 |
| 4-4 | Procedure for Backing Up the Program on a Memory Submodule | 4 - 7 |
| 4-5 | “STATUS” Test Function | 4 - 9 |
| 4-6 | “STATUS VAR” Test Function | 4 - 9 |


**Table**


| | | |
|---|---|---|
| 4-1 | Starting Up the Programable Controller | 4 - 4 |

# 4 Start-up and Program Tests

## 4.1 Operating Instructions

### 4.1.1 CPU Operator Panel

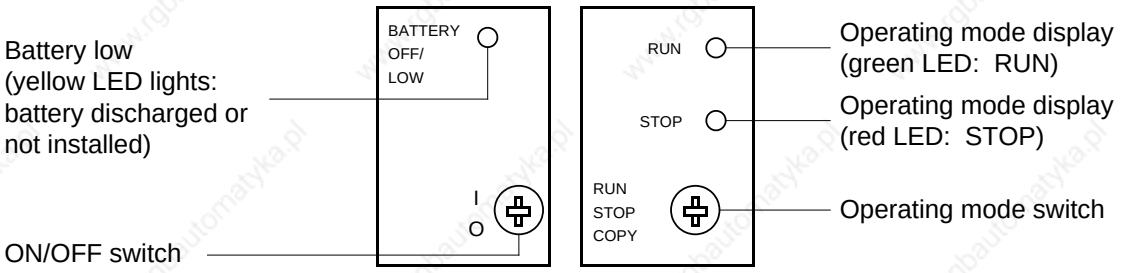



**Figure 4.1 CPU Operator Panel**

**ON/OFF Switch**

The ON/OFF switch turns on the CPU's voltage regulators. This switch does NOT separate the voltage regulator from the L+/M terminals.

**Operating Mode Switch**

Use the operating mode switch to select either the RUN or STOP operating mode. The CPU automatically goes into the START-UP mode during the transition from STOP to RUN (see section 7.4.2).

### 4.1.2 Operating Modes

**STOP Operating Mode**

- The program is not executed.
- The current values for timers, counters, flags, and process image I/O tables are saved when the STOP mode begins.
- The output modules are disabled (signal status "0").
- The process image I/O tables, timers, and non-retentive flags and counters are set to "zero" during the transition from STOP to RUN.

**RUN Operating Mode**

- The program is processed cyclically.
- Already started timers continue to run.
- The signal states for the input modules are stored.
- The output modules are addressed.
- The RUN operating mode can also be set after an OVERALL RESET, that is, when the program memory is empty.

**START-UP Operating Mode**

- The operating system processes DB1 and accepts the parameters (see section 9.1).
- Either the start-up organization block OB21 or OB22 is processed (see section 7.4.2).
- The amount of time start-up requires is not limited since the scan time monitor is not activated.
- Neither time-controlled program processing nor interrupt-driven program processing is possible.
- The input modules and output modules are disabled during start-up.

**Changing Operating Modes**

A change in operating mode can be caused by the following:

- The operating mode switch - when its position is changed.
- A programmer - if the operating mode switch on the programmable controller is set to RUN.
- Malfunctions - if one occurs that causes the programmable controller to go into the STOP operating mode (see chapter 5).

### 4.1.3 Performing an Overall Reset on the Programmable Controller

You should perform an overall reset before you input a new program. An overall reset erases the following:

- The programmable controller's program memory
- All data (flags, timers, and counters)
- All error IDs

> **Note**
>
> If you do not perform an overall reset, then the information indicated above is retained even if the program is overwritten.

**Manual Reset**

To perform a manual overall reset, you must:

1. Set the operating mode switch to STOP.
2. Remove the battery.
3. Set the ON/OFF switch to “0”.
4. Change the ON/OFF switch to “1”.
5. Insert the battery.

**Performing an Overall Reset with the Programmer**

You can select the overall reset function from the programmer's menu line. Refer to the programmer manual.

## 4.2 Starting Up a System

The following section contains suggestions for configuring and starting up a system containing programmable controllers.

### 4.2.1 Suggestions for Configuring and Installing the Product

A programmable controller is often used as a component in a larger system. The suggestions contained in the following warning are intended to help you safely install your programmable controller.

**Warning**

- Adhere to any safety and accident-prevention regulations applicable to your situation and system.
- If your system has a permanent power connection (stationary equipment) that is not equipped with an isolating switch and/or fuses that disconnect all poles, install either a suitable isolating switch or fuses in the building wiring system. Connect your system to a ground conductor.
- Before start-up, if you have units that operate using the main power supply, make sure that the voltage range setting on the equipment matches the local main power voltage.
- When using a 24 V supply, make sure to provide proper electric isolation between the main supply and the 24-V supply. Power supply units must meet the requirements of EN 60950 or be manufactured in accordance with DIN VDE 0551/EN 60742 and DIN VDE 0160. The requirements of electro-magnetic compatibility (EMC) must also be adhered to.
- Fluctuations or deviations of the supply voltage from the rated value may not exceed the tolerance limit specified in the technical data. If they do, functional failures or dangerous conditions can occur in the electronic modules or equipment.
- Take suitable measures to make sure that programs that are interrupted by a voltage dip or power failure resume proper operation when the power is restored. Make sure that dangerous operating conditions do not occur even momentarily. If necessary, force an EMERGENCY OFF.
- EMERGENCY OFF devices must be in accordance with EN 60204/IEC 204 (VDE 0113) and be effective in all operating modes of the equipment. Make certain to prevent any uncontrolled or undefined restart when the EMERGENCY OFF devices are released.
- Install power supply and signal cables so that inductive and capacitive interference can not affect the automation functions.
- Install your automation system and its operative components so as to prevent unintentional operation.
- Automation equipment can assume an undefined state in the case of a wire break in the signal lines. To prevent this, take the proper hardware and software safety measures when linking the inputs and outputs of the automation equipment.

## 4.2.2 Procedures for Starting Up the Programmable Controller

**Table 4-1. Starting Up the Programmable Controller**

| Prerequisites Procedures | Remarks | Displays |
|---|---|---|
| **System and programmable controller are off-load.**<br>• Check the mechanical configuration and wiring. (see section 3.1 and 3.2). | Check the mechanical assembly (VDE 0100 and VDE 0160). Terminal “M” of the load power supply and the ground terminal of the programmable controller must be connected to the central grounding point (standard mounting rail). For non-floating modules, a module’s “M” terminal must be connected to the programmable controller’s “M” terminal. | |
| **Set the ON/OFF switch to “0” and the operating mode switch to ”STOP”.** | | |
| • Switch on the power supply and load power supply.<br>• Set ON/OFF switch to “1”.<br>• Connect programmer to CPU.<br>• Reset the programmable controller (see section 4.1.3). | | • Red fault LEDs on the I/O modules lights.<br>• Red LED of the CPU lights;. yellow LED lights if the battery is low or not installed. |
| • Set operating mode switch to RUN. | | • Green LED on the CPU lights. |
| • Switch on sensor power supply. | | • Red fault LEDs on the input modules darken. |
| • Actuate the sensors one after the other. | The input signals in the PII can be observed with the “STATUS VAR” programmer function. | • Green LEDs on the input modules light. |
| • Switch on power supply for output modules and actuators. | | • Red fault LEDs on the output modules darken. |
| • Force the outputs with the “FORCE” programmer function. | The switching states of the associated actuators change. | • Green LEDs of the output modules light up. |
| **Program on memory submodule**<br>• Set ON/OFF switch to “0”.<br>• Plug in the memory submodule. | | |
| • Set ON/OFF switch to “1”. * | Program is loaded. | • Red LED of the CPU lights. |
| • Test program and make any necessary corrections.<br>• Set operating mode switch to STOP.<br>• Switch on the load.<br>• Set operating mode switch to RUN. | | |
| • **Back up the program.** | The system is in operation. | • Green LED of the CPU lights. |

* For the CPU 102 only: press the <COPY> key simultaneously (manual loading).

## 4.3 Loading the Program into the Programmable Controller

You can load a program from a connected programmer (online operation). When you load a program, it is transferred to the programmable controller's program memory. There are specific instructions in your programmer manual for doing this.

You can also load your program from a memory submodule, but only valid blocks can be loaded. See section 7.5.2. The different memory submodules you can use are listed in Appendix D. Section 4.3 describes how you can load a program from a memory submodule.

**Warning**

You can connect or disconnect memory submodules only in the Power OFF mode.

### Loading the Program Automatically

Automatic loading copies the program from a memory submodule into the program memory of the CPU. You can only load valid blocks. See section 7.5.2.

Figure 4-2 shows how a program can be loaded automatically.

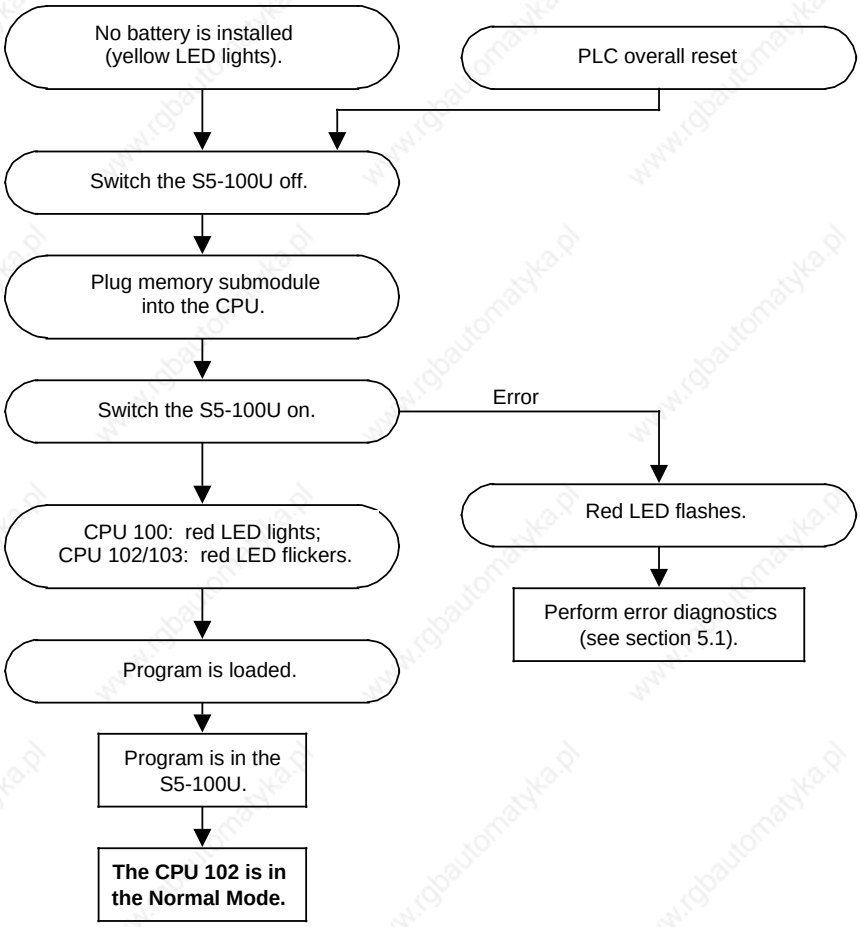



**Figure 4-2. Procedure for Loading the Program Automatically**

### Loading the Program Manually

Manual loading copies the program from a memory submodule into the program memory of the CPU. If a back-up battery is installed, any program in the memory is completely erased.

You can only load valid blocks. See section 7.5.2.

Figure 4-3 shows how a program can be loaded manually.

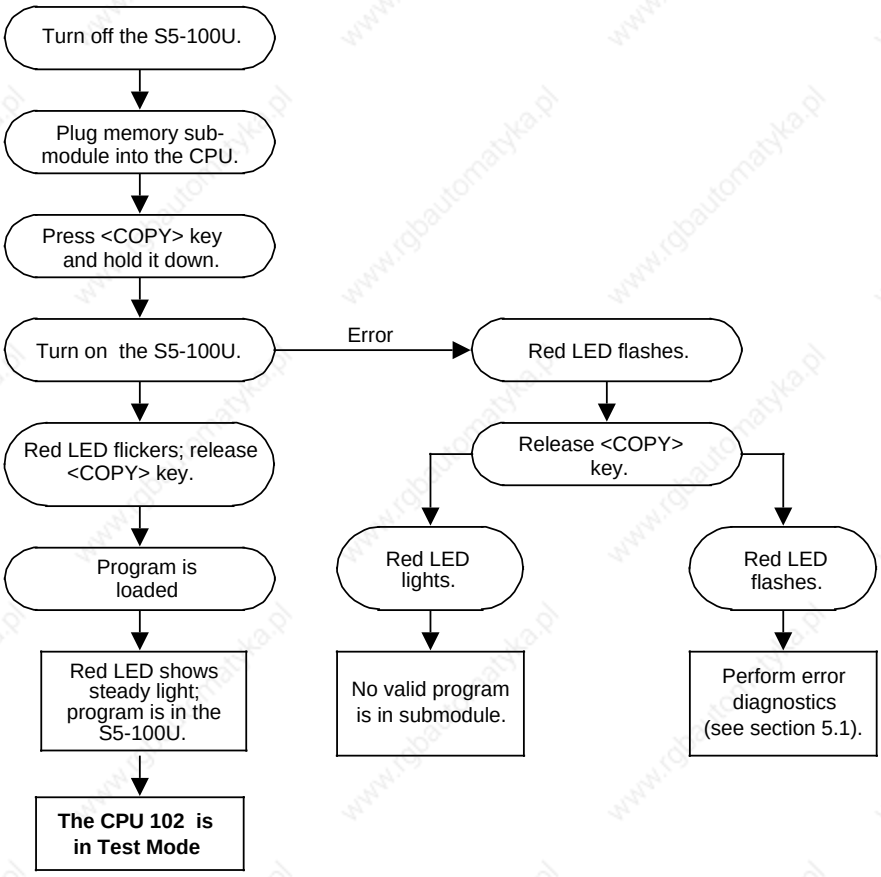


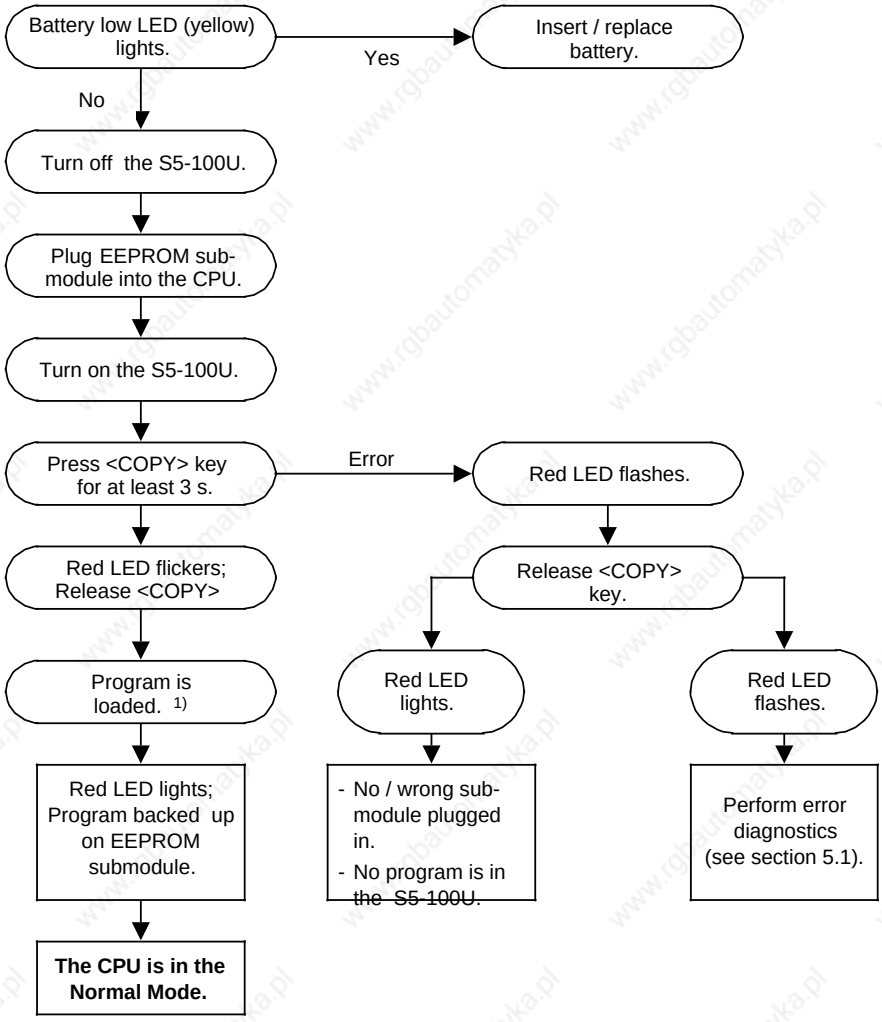

**Figure 4-3. Procedure for Loading the Program Manually**

## 4.4 Backing Up the Program

A program can be backed up only if the back-up battery is connected. Backing up copies a program from the program memory of the CPU to a memory submodule. Only valid blocks are backed up. As soon as you have changed the integral, default DB1 data block, it is a valid block that can be backed up. See section 7.5.2.

### 4.4.1 Backing Up the Program on a Memory Submodule

You can use various EEPROM memory submodules to back up a program. Appendix D contains a list of the submodules you may use. Figure 4-4 illustrates how to back up a program on a memory submodule.




1) Program load time: 40 s/1024 statements

**Figure 4-4. Procedure for Backing Up the Program on a Memory Submodule**

### 4.4.2 Function of the Back-Up Battery

If the power fails or the programmable controller is turned off, the contents of the internal (retentive) memory are stored only if a back-up battery is connected. When power is recovered or when the programmable controller is turned on, the following contents are available:

- Control program and data blocks (see section 7.3.5)
- Retentive flags and count values (see section 2.2.1)
- ISTACK contents (see section 5.3]

**Note**

- Insert and replace the battery while the programmable controller is turned on. Otherwise, an OVERALL RESET is required when you turn the programmable controller on.
- The lithium battery in the programmable controller has a life expectancy of at least one year.
- The yellow LED on the operator panel lights up if the battery fails.

**Warning**

Do not charge lithium batteries. They could explode. Dispose of used batteries properly.

## 4.5 Program-Dependent Signal Status Display “STATUS”

This test function displays the current signal states and the Result of Logic Operations (RLO) of the individual operands during program processing.
You can use this test function to make corrections to the program.

**Note**

The current signal states are displayed only in the RUN operating mode.

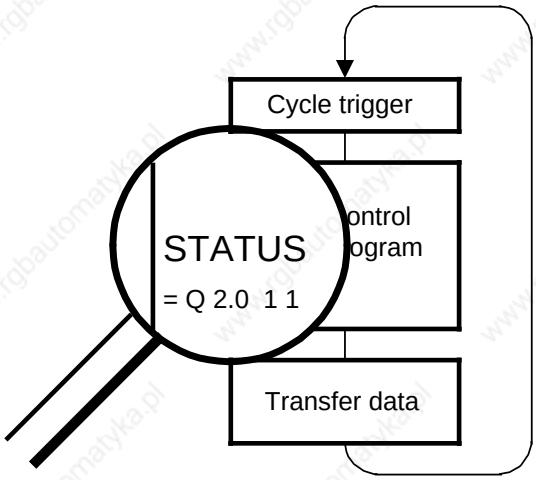



**Figure 4-5. "STATUS" Test Function**

Refer to your programmer manual for information about the test function on your programmer.

## 4.6 Direct Signal Status Display "STATUS VAR"

This test function specifies the status of the operands (inputs, outputs, flags, data words, counters, or timers) at the end of program processing. You can obtain information about inputs and outputs from the process image I/O tables of the selected operands.

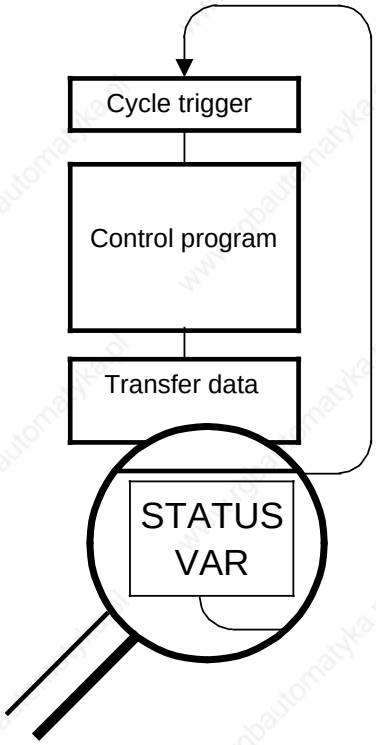



**Figure 4-6. "STATUS VAR" Test Function**

Refer to your programmer manual for information about the test function on your programmer.

## 4.7 Forcing Outputs, “FORCE”, for CPU 103 and Higher

Outputs can be set directly to a desired status even without the control program. This enables you to control the wiring and functionality of output modules. This does not change the process I/O image table, but the output disable condition is cancelled.

> **Note**
>
> The programmable controller must be in the STOP operating mode.

Refer to your programmer manual for information about calling up the test function on your programmer.

## 4.8 Forcing Variables, “FORCE VAR”

The process image I/O table of the operands is changed regardless of the programmable controller's operating mode. You can change the following variables: I, Q, F, T, C, and D.
The program is processed in the RUN operating mode using the changed process variables. They can be changed again during program scanning without an acknowledgement being required. The process variables are forced asynchronously to the program scanning.

**Special characteristics**

- You can change the I, Q, and F variables in the process I/O image table by bits, bytes, or words.
- For the T and C variables in KM and KH format, note the following:
  - For programmers with screens, you must also enter “YES” in the system commands input field in the presettings screen.
  - You must be careful when you force edge trigger flags. You do not want to enable a higher-order byte inadvertently because this could give you a timer or counter value you did not set.
- The signal status display breaks off if there is an error in the format entry or operand entry. The programmer then displays the “NO FORCING POSSIBLE” message.

Refer to your programmer manual for information about the test function on your programmer.

## 4.9 Search Function

This function allows you to search for specific terms in the program and list them on the programmer's display panel. You can perform program changes at this point.

You can have search runs in the following programmer functions:

- INPUT
- OUTPUT
- STATUS

Some of the items you can search for are:

- Statements (e.g., A I 0.0)
- Operands (e.g., Q 3.5)
- Labels (e.g., X 01); possible only in function blocks
- Addresses (e.g., $0006_H$)

> **Note**
>
> Search runs are handled differently by different programmers. The respective users guides contain extensive information about search runs.

## 4.10 Program Check, for CPU 103 and Higher

When this programmer function is called up, program scanning is stopped at a definite point. The cursor indicates this breakpoint, which is a statement in the program. The programmable controller scans the program up to the statement selected. The current signal states and the RLO up to the statement selected are displayed (as in the “STATUS” test function).

The program can be scanned section by section by shifting the breakpoint. Program scanning takes place as follows:

- All jumps in the block called are executed.
- Block calls are executed immediately. The program check is not resumed until control is returned to the calling block.

The following applies during the program check:

- The two mode LEDs are not lit.
- The program writes to the PIQ and reads out the PII.
- No process image (data cycle) is transferred.
- All outputs are set to zero.

During the program check, you can execute the following additional test and programmable controller functions from the programmer:

- Input and output (program modification possible)
- Direct signal status display (STATUS VAR)
- Forcing of outputs and variables (FORCE, FORCE VAR)
- Information functions (ISTACK, BSTACK)

If the function is aborted due to hardware faults or program errors, the programmable controller goes into the STOP mode and the red LED on the control panel of the CPU lights.

Refer to your programmer manual for information about calling up these functions on your programmer.

## 5 Diagnostics and Troubleshooting


| | | |
|---|---|---|
| 5.1 | Indication of Errors by LEDs | 5 - 1 |
| 5.2 | CPU Malfunctions | 5 - 1 |
| 5.2.1 | “ISTACK” Analysis Function | 5 - 1 |
| 5.2.2 | Interrupt Analysis | 5 - 4 |
| 5.2.3 | Errors during Program Copying | 5 - 5 |
| 5.2.4 | Explanation of the Mnemonics Used in “ISTACK” | 5 - 6 |
| 5.3 | Program Errors | 5 - 8 |
| 5.3.1 | Locating the Error Address | 5 - 8 |
| 5.3.2 | Tracing the Program with the “BSTACK” Function | 5 - 11 |
| 5.4 | I/O Faults | 5 - 12 |
| 5.5 | System Parameters | 5 - 12 |
| 5.6 | The Last Resort | 5 - 13 |

**Figures**


5-1 Structured Program with an Illegal Statement . . . . . 5 - 8
5-2 Addresses in the CPU's Program Memory . . . . . 5 - 9
5-3 Calculating the Error Address . . . . . 5 - 10
5-4 Tracing the Program with "BSTACK" . . . . . 5 - 11
5-5 Analyzing the Cause of a Fault in the I/Os . . . . . 5 - 12


**Tables**


5-1 Error Indication and Error Analysis . . . . . 5 - 1
5-2 ISTACK Output (Bytes 1 to 16) . . . . . 5 - 2
5-3 Interrupt Analysis . . . . . 5 - 4
5-4 Errors when Copying . . . . . 5 - 5
5-5 Meaning of the Remaining ISTACK Bits . . . . . 5 - 6
5-6 Mnemonics Used for the Interrupt Display . . . . . 5 - 7

# 5 Diagnostics and Troubleshooting

## 5.1 Indication of Errors by LEDs

The programmable controller's operator panel will show you if your device is not functioning correctly (see Table 5-1).

**Table 5-1. Error Indication and Error Analysis**

| Error Indication | Error Analysis |
|---|---|
| CPU in STOP<br>Red LED lights | CPU malfunction<br>Use the programmer to execute an interrupt analysis<br>(see section 5.2). |
| CPU in STOP<br>Red LED flashes | Error when loading or backing up the program<br>Use the programmer to execute an interrupt analysis<br>(see section 5.2). |
| CPU in RUN<br>Green LED lights<br>Faulty operation | Program error<br>(see section 5.3)<br>or<br>I/O fault<br>Execute a fault analysis<br>(see section 5.4). |

If both LEDs light, your programmable controller is in the START-UP operating mode.

## 5.2 CPU Malfunctions

### 5.2.1 “ISTACK” Analysis Function

The interrupt stack is an internal CPU memory area where the causes of malfunctions are stored. If there is a malfunction, a bit in the respective byte of the memory area is set. Using the programmer, you can read out the contents of this memory area byte-by-byte.

**Calling the ISTACK**

The call is made through the programmer menu in the STOP operating mode.
Refer to your programmer manual for the key sequence.

> **Note**
>
> Only ISTACK bytes 1 through 6 can be output in the RUN mode. There is no cause for an interrupt to force the CPU to go into the STOP mode. The control bits are output in bytes 1 through 6.

The following table shows which positions in the bit pattern are relevant for error diagnosis (gray-shaded bits).

**Table 5-2. ISTACK Output (Bytes 1 to 16)**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Absolute Addr. | Syst. Data Word (SD) |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | BST SCH | SCH TAE | ADR BAU | | | | EA0A | SD 5 |
| 2 | | | | | | | | | | |
| 3 | STO ZUS | STO ANZ | NEU STA | | BAT PUF | | | | EA0C | SD 6 |
| 4 | | | | | | AF | | | | |
| 5 | | | KOPF NI | | | | | | EA0E | SD 7 |
| 6 | KEIN AS | SYN FEH | NINEU | | | | | UR LAD | | |
| 7 | IRRELEVANT | | | | | | | | | |
| 8 | IRRELEVANT | | | | | | | | | |
| 9 | STOPS | | SUF | TRAF | NNN | STS | STUEB | | EBAC | SD 214 (UAW) |
| 10 | NAU | | | ZYK | SYSFE | PEU | BAU | ASPFA | | |
| 11 | | | | | | | | | EBAA | SD 213 |
| 12 | ANZ1 | ANZ0 | OVFL | | OR | STA TUS | VKE | ERAB | | |
| 13 | 6th nesting level | | | | | OR | VKE | FKT | EBA8 | SD 212 |
| 14 | IRRELEVANT | | | | | | | | | |
| 15 | 4th nesting level | | | | | OR | VKE | FKT | EBA6 | SD 211 |
| 16 | 5th nesting level | | | | | OR | VKE | FKT | | |

**Table 5-2. ISTACK Output (Bytes 17 to 32) [continued]**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Absolute Addr. | Syst. Data Word (SD) |
|---|---|---|---|---|---|---|---|---|---|---|
| 17 | 2nd nesting level | | | | | OR | VKE | FKT | EBA4 | SD 210 |
| 18 | 3rd nesting level | | | | | OR | VKE | FKT | | |
| 19 | Nesting depth (0 to 6) | | | | | | | | EBA2 | SD 209 |
| 20 | 1st nesting level | | | | | OR | VKE | FKT | | |
| 21 | Start address of the data block (high) | | | | | | | | EBA0 | SD 208 |
| 22 | Start address of the data block (low) | | | | | | | | | |
| 23 | Block stack pointer (high) | | | | | | | | EB9E | SD 207 |
| 24 | Block stack pointer (low) | | | | | | | | | |
| 25 | Step address counter (high)* | | | | | | | | EB9C | SD 206 |
| 26 | Step address counter (low)* | | | | | | | | | |
| 27 | Operation register (high) | | | | | | | | EB9A | SD 205 |
| 28 | Operation register (low) | | | | | | | | | |
| 29 | ACCU 2 (high) | | | | | | | | EB98 | SD 204 |
| 30 | ACCU 2 (low) | | | | | | | | | |
| 31 | ACCU 1 (high) | | | | | | | | EB96 | SD 203 |
| 32 | ACCU 1 (low) | | | | | | | | | |

* The absolute memory address of the next statement to be processed from the faulty block is displayed. If the step address counter displays a DB1 address, then there is a DB1 parameter setting error (see section 9.1).

## 5.2.2 Interrupt Analysis

When there is an interrupt in program processing, you can use the following table to determine the cause of the error. The CPU always goes into the STOP mode.

**Table 5-3. Interrupt Analysis**

| ISTACK Display | Byte | Cause of Error | Remedy |
|---|---|---|---|
| **ASPFA** and **KEIN AS** and **NNN** and SAZ=FFFF* (CPU 102) | 10<br>6<br>9<br>25 and 26 | Error during program transfer from the PG to the PLC:<br>Overflow of the internal program memory during compilation | Shorten program.<br>Compress memory. |
| **BAU** | 10 | When automatically loading the program:<br>- Battery is missing or dead and there is no valid program available on the memory submodule | Replace the battery and recreate the program, or load the program again. |
| **NAU** | 10 | Interruption in the power supply voltage to the CPU | |
| **NINEU** | 6 | The program in the PLC memory is defective.<br>Cause:<br>• A power failure has interrupted one of the following operations.<br>- Compress<br>- Block transfer from the PG to the PLC or memory submodule to the PLC<br>- PLC overall reset<br>• Battery has been replaced while the power was off. | Perform an overall reset and load the program again. |
| **NNN** | 9 | • Statement cannot be decoded.<br>• Nesting level is too high.<br>• Parameter exceeds permitted limits. | Eliminate program errors. |
| **PEU** | 10 | • Expansion module not connected<br>• I/O bus malfunction<br>• Maximum length of shift register exceeded<br>• Module unknown<br>• Module in wrong slot | • Check the power supply in the expansion unit.<br>• Check the connections.<br>• Check the module slots. |
| **STOPS** | 9 | Operating mode switch on STOP | Set to RUN |
| **STS** | 9 | • Software stop by statement (STP)<br>• STOP requested by programmer | |
| **STUE** | 9 | Block stack overflow: the maximum block call nesting depth (16) has been exceeded. | Eliminate program errors. |
| **SYS** FEH** | 10 | DB1 parameter setting error | Correct DB1. |

* SAZ = STEP address counter - The ISTACK bytes 25 and 26 read “1111 1111(FF)”.
** Relevant only for the PG 605U and for the CPU 103, version 8MA03 and higher.

**Table 5-3. Interrupt Analysis (continued)**

| ISTACK Display | Byte | Cause of Error | Remedy |
|---|---|---|---|
| **SUF*** | 9 | Substitution error:<br>Function block called with an incorrect actual parameter | Change actual parameter. |
| **TRAF** | 9 | Transfer error<br>• Data block statement programmed with a data word number larger than the data block length<br>• Data block statement programmed without previously opening a data block | Eliminate program error (see your programmer manual). |
| **ZYK** | 10 | Scan time exceeded:<br>The program processing time exceeds the set monitoring time. Causes:<br>• Program too long<br>• Interrupts too frequent | Check the program for continuous loops or shorten program. |

* Relevant for CPU 102, version 8MA02 and higher

### 5.2.3 Errors during Program Copying

Error message: after the <COPY> key is released, the red LED continues flashing.

**Table 5-4. Errors when Copying**

| ISTACK Display | Cause of Error | Remedy |
|---|---|---|
| **ASPFA** | Loading the memory submodule into the PLC:<br>• Program on the memory submodule is too long for the PLC's program memory.<br>• Program on the module contains an invalid block number. | Check the program on the memory submodule. |
| **ASPFA** | Saving from the PLC to the memory submodule:<br>EEPROM memory submodule is defective or too small for the program in the PLC memory. | Replace the memory submodule, or use a larger EEPROM memory submodule. |
| **ASPFA** and **KEIN AS** and **NNN** and SAZ=FFFF* (CPU 102) | Internal program memory overflow during compilation | Shorten program. |

* SAZ = STEP Address Counter
The ISTACK bytes 25 and 26 read “1111 1111(FF)”

### 5.2.4 Explanation of the Mnemonics Used in “ISTACK”

**Table 5-5. Meaning of the Remaining ISTACK Bits**

| ISTACK Display | Byte | Explanation |
|---|---|---|
| **BST SCH**<br>**SCH TAE**<br>**ADR BAU** | 1 | Shift block.<br>Execute shift operation.<br>Structure address list. |
| **STO ANZ**<br>**STO ZUS**<br>**BAT PUF**<br>**NEU STA** | 3 | PLC in STOP<br>Internal control bit for STOP/RUN change<br>Battery backup available<br>PLC not yet in cycle after Power ON<br>- See bytes 9 and 10 for cause. |
| **AF*** | 4 | Interrupt enable/enabling of time-controlled OB13 and interrupt-driven OB3 |
| **KOPFNI** | 5 | Program contains errors.<br>Block header cannot be interpreted. |
| **KEIN AS**** | 6 | Not enough S5 statement memory available |
| **URLAD**<br>**SYNFEH** | | Overall reset, program defective<br>Program contains errors. |
| **ANZ 1/ANZ 0** | 12 | Condition code bits for arithmetic, logic, and shift operations. |
| **OV** | | Arithmetic overflow |
| **OR** | | ID bit of OR memory |
| **STATUS** | | Status ID of operand of last binary statement executed |
| **VKE** | | Result of logic operation (RLO) |
| **ERAB** | | ID bit of first scan |
| **FKT** | 13 | 0: O( OR parenthesis open<br>1: A( AND parenthesis open |

* relevant for CPU 103 only
** for CPU 102: 0 = normal mode
1 = test mode

**Table 5-6. Mnemonics Used for the Interrupt Display**

| Mnemonics Used for the Interrupt Display | Explanation |
|---|---|
| ANZ1/ANZ0 | Condition codes for various operations (see section A.1.4) |
| ASPFA | Illegal memory submodule |
| BAU | Battery failure |
| ERAB | First scan |
| FKT | 0 : O( 1 : A( |
| KE1...KE6 | Nesting stack entry 1 to 6 entered for A( and O( |
| KEINAS | Insufficient S5 statement memory available |
| NAU | Power failure |
| NINEU | Cold restart not possible |
| NNN | Statement cannot be interpreted in the PLC |
| OR | OR memory (set by command “0”) |
| OVFL | Arithmetic overflow (+ or -) |
| PEU | I/Os not ready:<br>• First bus unit not connected<br>• Expansion module not connected<br>• I/O bus malfunction<br>• Maximum shift register length exceeded<br>• Unknown module<br>• Module in the wrong slot |
| STATUS | STATUS of the operand of the last binary statement executed |
| STOPS | Operating mode switch on STOP |
| STS | Operation interrupted by a programmer STOP request or programmed STOP statements |
| STUE | Block stack overflow: The maximum block call nesting depth of 16 has been exceeded. |
| SUF | Substitution error |
| SYSFEH* | Error in DB1 |
| TRAF | Transfer error for data block statements:<br>• When accessing a data word even though no corresponding data block was opened or<br>• When the data word number is larger than the data block length |
| UAW | Interrupt display word |
| VKE | Result of logic operation (RLO) |
| ZYK | Scan time exceeded: the set maximum permissible program scan time has been exceeded |

* Relevant only for CPU 103 version 8MA03 and higher

## 5.3 Program Errors

### 5.3.1 Locating the Error Address

The SAZ (STEP address counter) in the ISTACK (bytes 25 and 26) contains the absolute address of the STEP 5 statement in the programmable controller **before** which the CPU went into the STOP mode.

Use the “DIR PC” programmer function to determine the associated block start address.

**Example**: You have entered a control program consisting of OB1, PB0 and PB7. An illegal statement has been programmed in PB7.

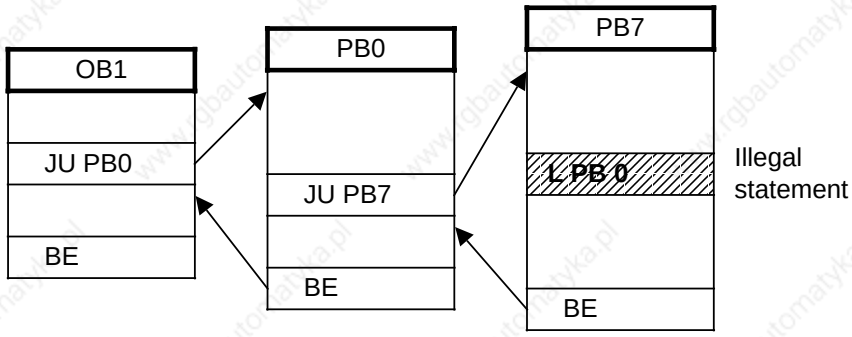



**Figure 5-1. Structured Program with an Illegal Statement**

When it reaches the illegal statement, the CPU interrupts program scanning and enters the STOP mode with the “NNN” message. The STEP address counter is at the absolute address of the next (but not yet scanned) statement in the program memory.

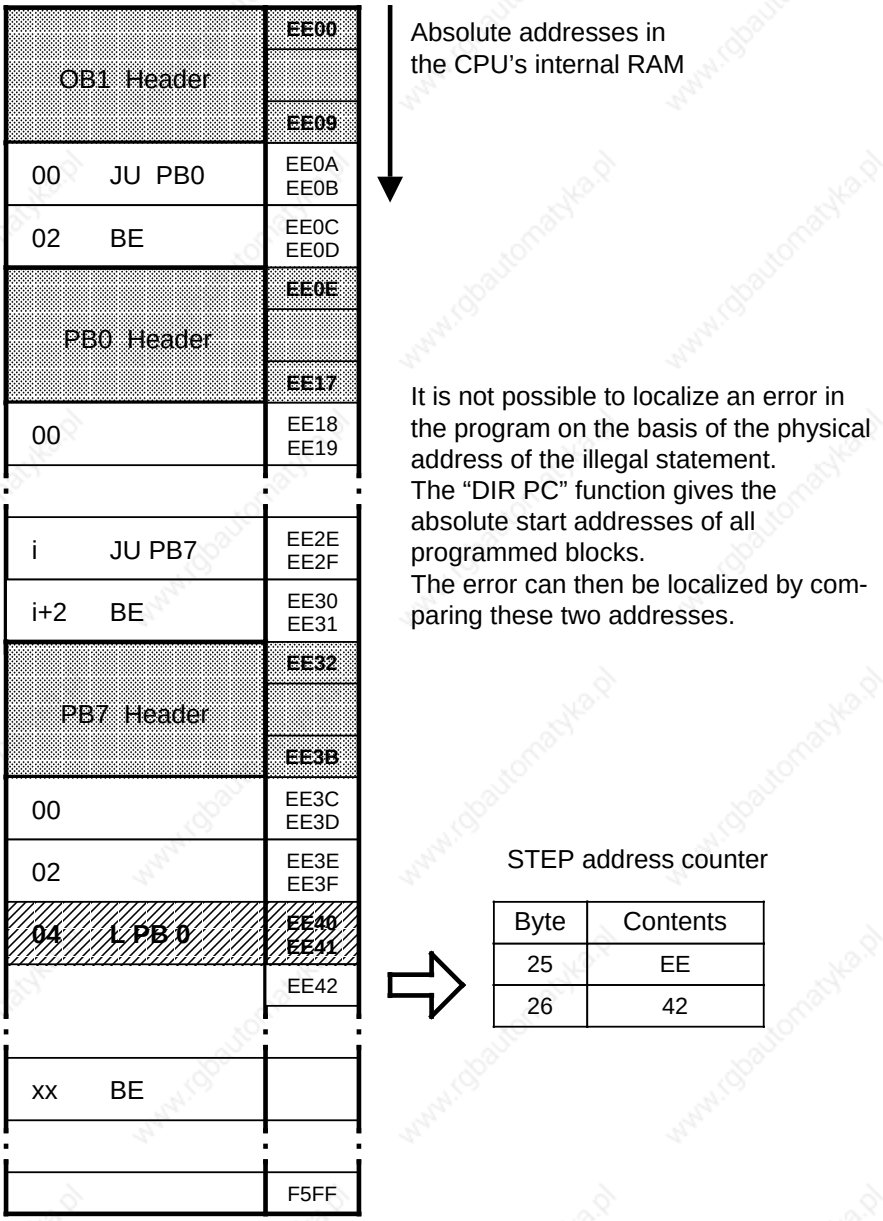



**Figure 5-2. Addresses in the CPU's Program Memory**

**Calculating the Address (necessary only when using the PG 605U)**

In order to be able to make program corrections, it is necessary to have the address of the statement that led to the fault referenced to the particular block (relative address).
The faulty block is found by comparing the SAZ (STEP address counter) contents and the "DIR PC" display.
The relative error address gives the difference between the SAZ value and the block start address.
Figure 5-3 gives you an example of how to calculate the relative error address.

| **ISTACK byte** | 25 | 26 |
|---|---|---|
| **STEP address counter** | EE | 42 |

The absolute address EE42 is greater than the start address for PB7. The faulty statement is therefore in PB7.

| **DIR PC** | |
|---|---|
| Block | Start Address |
| PB0 | EE18 |
| PB7 | EE3C |
| OB1 | EE0A |

**Calculating the relative address:** EE42 - EE3C = 0006

"0006" is the relative address of the statement in PB7 **following** the statement that caused the CPU to go into the STOP mode.

**Figure 5-3. Calculating the Error Address**

**Output of an Error Statement**

Use the "SEARCH" programmer function to find certain program locations and to look for the relative error address. Refer to your programmer manual for additional information about this programmer function.

## 5.3.2 Tracing the Program with the “BSTACK” Function

Program trace with “BSTACK” is not possible on the 605U programmer.

During program processing, the following information about jump operations is entered in the block stack (BSTACK):

- The data block that was valid before program processing exited a block.
- The relative return address
  - It specifies the address where program processing will continue after the return from the called up block.
- The absolute return
  - It specifies the memory address in the program memory where program processing will continue after the return.

You can call up this information with the “BSTACK” programmer function in the STOP operating mode if a fault caused the CPU to go into the STOP operating mode. “BSTACK” then reports the status of the block stack at the time the interruption occurred.

**Example:**

Program scanning was interrupted at function block FB2. The CPU went into the STOP mode with the error message “TRAF” (because of incorrect DB access, e.g., DB5 is two words long and DB3 is ten words long).

“BSTACK” lets you determine the path used to reach FB2 and lets you know which DB was open at the time of call up. “BSTACK” contains the three (marked) return addresses.

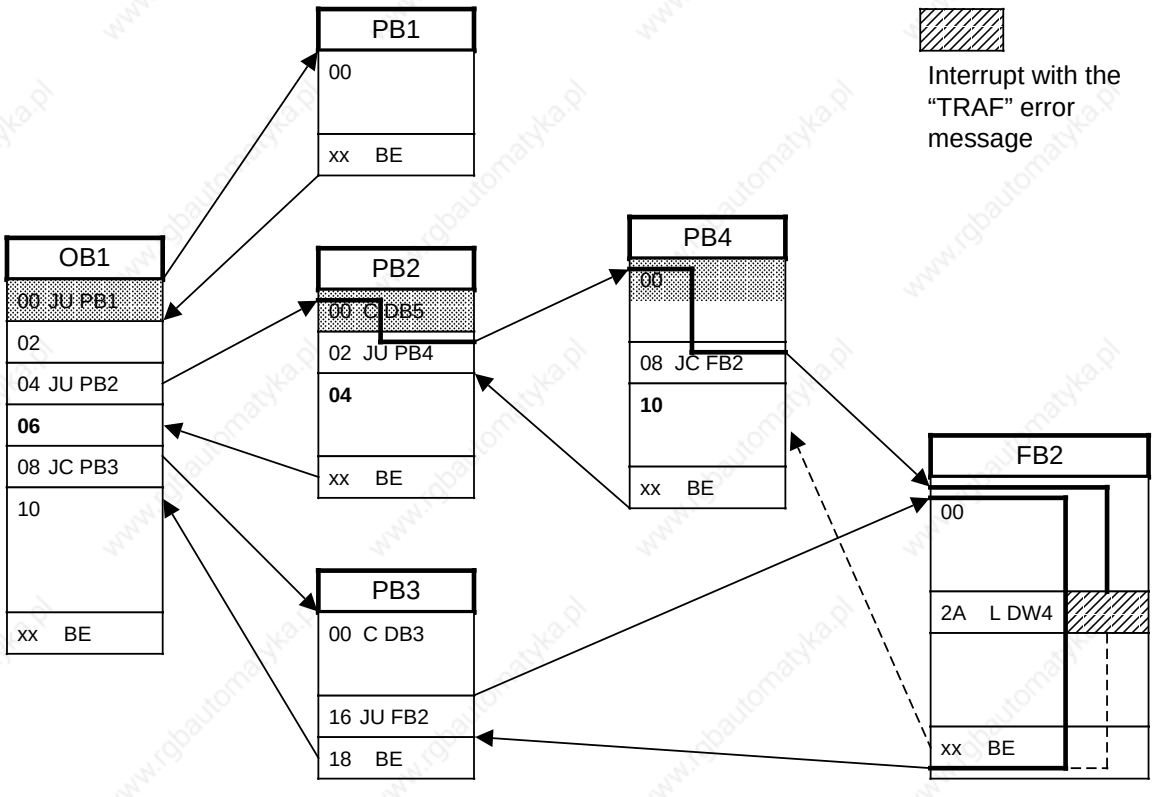



**Figure 5-4. Tracing the Program with “BSTACK”**

## 5.4 I/O Faults

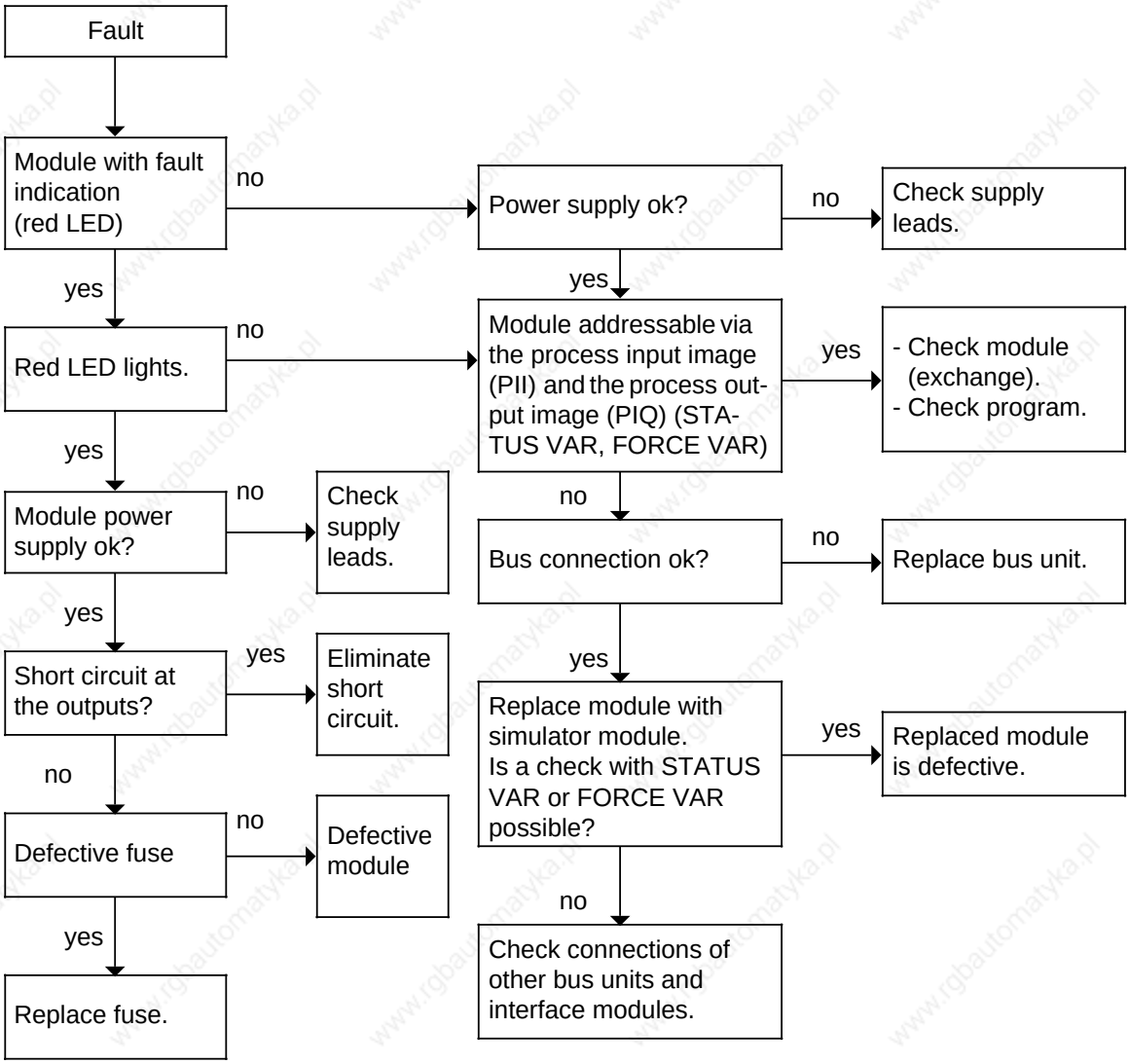



**Figure 5-5. Analyzing the Cause of a Fault in the I/Os**

## 5.5 System Parameters

The “SYSPAR” programmer function makes it possible to read out the system parameters (e.g., CPU software version) of the programmable controller (see programmer manual).

## 5.6 The Last Resort

The programmable controller will not go back to the RUN operating mode:

**Possible cause:** The battery was installed or changed when the programmable controller was turned off.

**Remedy:** Perform an overall reset and load the program again.

**How to perform an overall reset without a programmer**

1. Set the operating mode switch to STOP.
2. Remove the battery.
3. Set the ON/OFF switch to “0”.
4. Set the ON/OFF switch to “1”.
5. Install a battery.

Contact your local Siemens representative if the above measures are ineffective.

## 6 Addressing


| | | |
|---|---|---|
| 6.1 | Slot Numbering | 6 - 1 |
| 6.2 | Digital Modules | 6 - 4 |
| 6.3 | Analog Modules | 6 - 5 |
| 6.4 | Combined Input Modules and Output Modules | 6 - 6 |
| 6.4.1 | Output Modules with Error Diagnostics | 6 - 6 |
| 6.4.2 | Digital Input/Output Module, 16 Inputs, 16 Outputs, 24 V DC for All CPUs Version 8MA02 and Higher and for CPU 102, Version 8MA01, Revision 5 and Higher | 6 - 7 |
| 6.4.3 | Function Modules | 6 - 7 |
| 6.5 | The Structure of Process Image Input and Output Tables | 6 - 8 |
| 6.5.1 | Accessing the Process Image Input Table (PII) | 6 - 10 |
| 6.5.2 | Accessing the Process Image Output Table (PIQ) | 6 - 11 |
| 6.6 | Interrupt Process Images and Time-Controlled Program Processing in OB13 for CPU 103, Version 8MA02 and Higher | 6 - 12 |
| 6.6.1 | Accessing the Interrupt PII | 6 - 12 |
| 6.6.2 | Accessing the Interrupt PIQ | 6 - 14 |
| 6.7 | RAM Address Assignments | 6 - 15 |

**Figures**


6-1 Address Assignment . . . . . 6 - 1
6-2 Consecutive Numbering of Slots in a Single-Tier Configuration . . . . . 6 - 1
6-3 Slot Numbering in a Multi-Tier Configuration . . . . . 6 - 2
6-4 Expanding from 14 to 18 Slots . . . . . 6 - 3
6-5 Configuration of a Digital Address . . . . . 6 - 4
6-6 Address Assignment for Analog Modules . . . . . 6 - 5
6-7 Assignment of Process Images to the I/O Modules . . . . . 6 - 9
6-8 Accesses to the PII . . . . . 6 - 10
6-9 Accesses to the PIQ . . . . . 6 - 11
6-10 Accesses to the Interrupt PII . . . . . 6 - 13
6-11 Accesses to the Interrupt PIQ . . . . . 6 - 14


**Tables**


6-1 Error Messages for Output Modules with Error Diagnostics . . . . . 6 - 6
6-2 Address Assignment . . . . . 6 - 7
6-3 Structure of the PII and the PIQ . . . . . 6 - 8
6-4 Structure of the Interrupt PII and the Interrupt PIQ . . . . . 6 - 12
6-5 Important Addresses in the RAM . . . . . 6 - 15
6-6 System Data Area Assignment . . . . . 6 - 16

# 6 Addressing

The inputs and the outputs have different assigned addresses so that you can access them specifically. The I/O addresses are the same as the module slot addresses.

When you mount a module in a slot on a bus unit, the module is assigned a slot number and consequently a fixed byte address in one or both process image I/O tables.

Connect the sensors and actuators to the terminal block. The terminal selected determines the channel number.

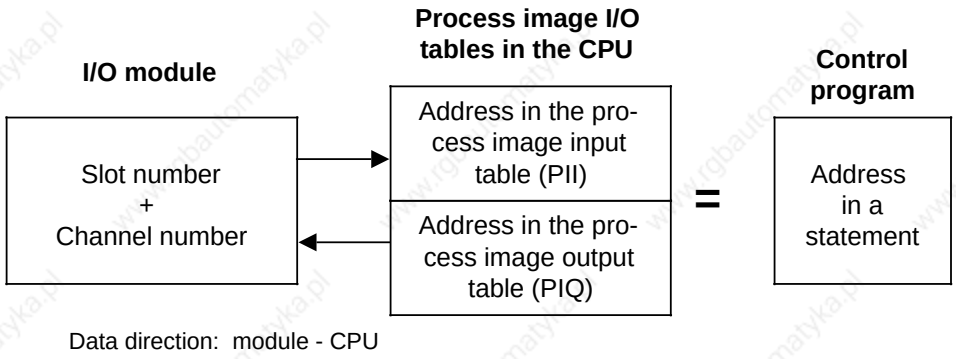



**Figure 6-1. Address Assignment**

## 6.1 Slot Numbering

The programmable controller can have a maximum of four tiers. You can use up to 16 bus units (32 slots). The slots are numbered consecutively. Numbering begins with “0” at the slot beside the CPU. Whether a module is plugged in or not has no effect on the numbering.

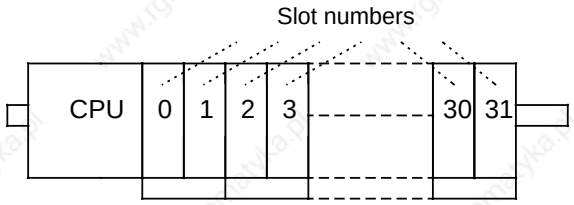



**Figure 6-2. Consecutive Numbering of Slots in a Single-Tier Configuration**

If the programmable controller consists of more than one tier, numbering of the expansion tiers is continued at the slot on the extreme left.

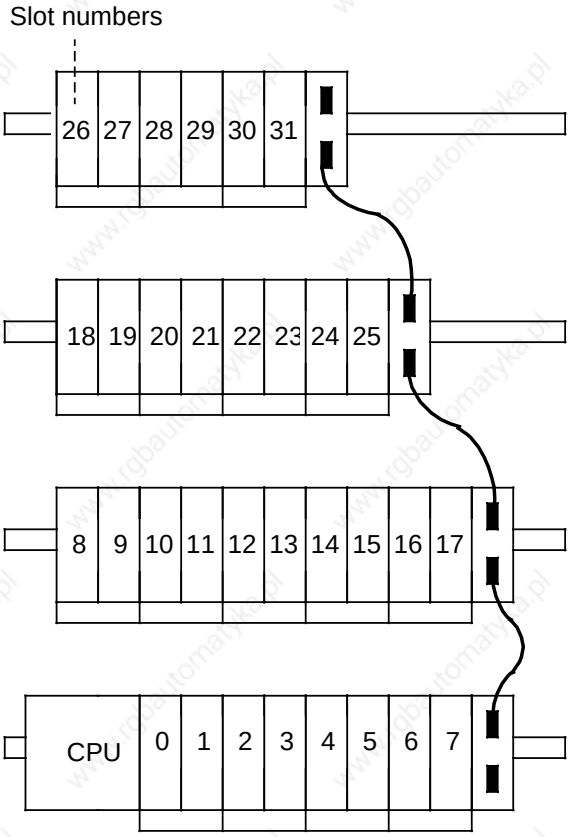



**Figure 6-3. Slot Numbering in a Multi-Tier Configuration**

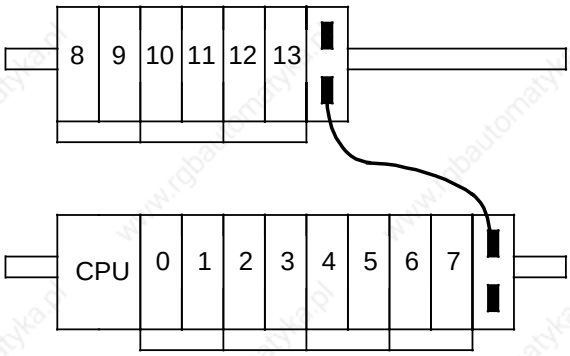
When expanding your system, always add the new bus units to the topmost tier on the right. Otherwise, the slot numbers on the right of the new bus units will be changed, requiring address changes in your control program.

**Note**

After every expansion, check to make certain that the addressing used in the control program is the same as that in the actual configuration.

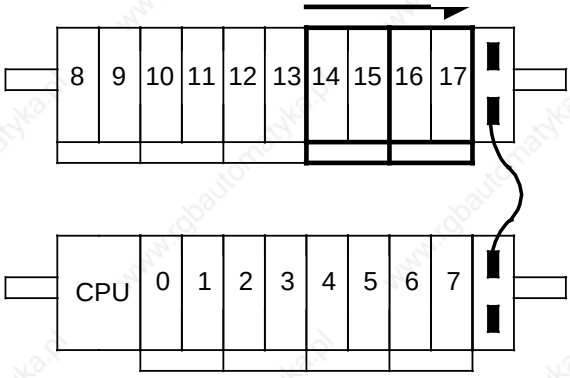
**Example:** Expanding from 14 to 18 slots

**Existing configuration**



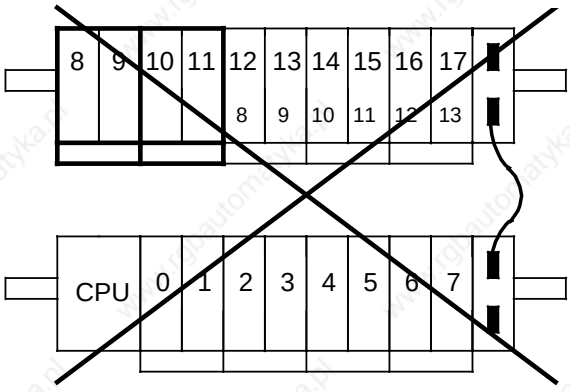
**New bus units**

**Correct expansion procedure**




The new bus units are added at the right. The interface module is moved correspondingly to the right. The old slot numbers are retained. Continue numbering the new slots sequentially.

**Incorrect expansion procedure**




The slot numbers of the old bus units move to numbers 12 to 17. The new slots are given the numbers 8 to 11.

**Figure 6-4. Expanding from 14 to 18 Slots**

## 6.2 Digital Modules

Digital modules can be plugged into all slots (0 through 31).
Only two information states (“0” or “1”, OFF or ON) per channel can be transferred from or to a digital module. The memory requirement is one bit.

Each channel of a digital module is displayed by a bit. This is the reason that every bit must be assigned its own number. Use the following form for a digital address:

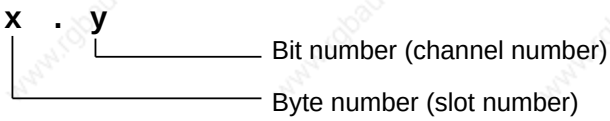



**Figure 6-5. Configuration of a Digital Address**

The “**X.Y**” address consists of the following two components:

- **Byte Address X (Slot Number X)**
  - The byte address is the same as the number of the slot the module is plugged into.
- **Channel Number Y (Bit Address Y)**
  - The channel number comes from the connection of the actuators or sensors to the terminals of the terminal block. The assignment for the channel number and the terminal number is printed on the frontplate of the module.

**Example:** Address Assignment

You are connecting a 2-wire BERO proximity limit switch to an 8 x 24-V DC digital input module (6ES5 421-8MA11) at terminal 3. The other wire is routed to an L+ (positive supply voltage) terminal block (see section 3.2 for wiring). The module is plugged into slot 3.

This defines the address used by the control program to evaluate the signal states of the BERO.

- The byte address is 3 since the module is plugged into slot 3.
- As shown on the frontplate, channel number 1 is used.
- The complete address for the BERO switch is 3.1.

### Note

You can address 4-channel digital modules only with channel numbers 0 through 3. The channel numbers 4 through 7 printed on the frontplate are relevant only for the ET 100U system.

## 6.3 Analog Modules

You can plug analog modules only into slots 0 through 7. Transfer of 65,536 different items of information is possible per channel from or to an analog module. The memory requirement is 16 bits=2 bytes=1 word. The modules are addressed byte-by-byte or word-by-word with load or transfer operations.

The programmable controller takes this increased address requirement into account when an analog module is plugged in.

- Eight bytes (=four words) are reserved per slot.
- Two bytes (=1 word) are reserved per channel.
- The slot addressing area is changed.
- The permissible address space extends from byte 64 (slot 0, channel 0) to byte 127 (slot 7, channel 3).

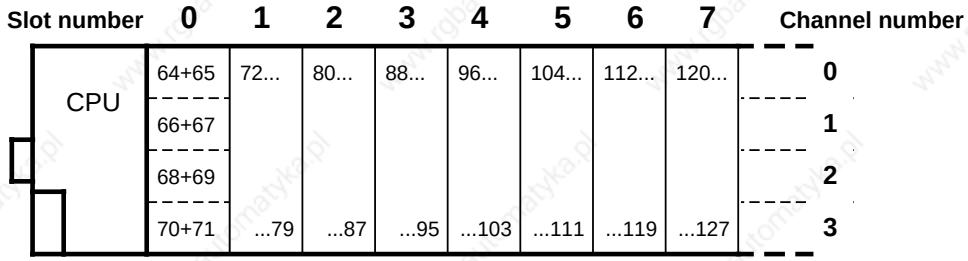



**Figure 6-6. Address Assignment for Analog Modules**

**Examples:** 1) Bytes 88+89=analog module in slot 3, channel number 0

2) Channel 1 address of an analog module in slot 5?
Solution: bytes 106+107

> **Note**
>
> Any combination of analog and digital modules is possible in slots 0 through 7.

## 6.4 Combined Input Modules and Output Modules

With these modules it is possible to write data from the control program to the module and to read in data from the module to the control program.

The byte addresses in the process image input table (PII) and process image output table (PIQ) are identical. The meaning of the transferred data is usually different.

### 6.4.1 Output Modules with Error Diagnostics

In addition to the fault LED (red LED), the following output modules can signal errors to the CPU.

| | |
|---|---|
| 4 x 24 V DC / 0.5 A | (6ES5 440-8MA12) |
| 4 x 24 V DC / 2.0 A | (6ES5 440-8MA22) |
| 4 x 24 to 60 V DC / 0.5 A | (6ES5 450-8MB11) |

You can read the error messages on input channels I X.0 and I X.1 (not with CPU 100, version 8MA01).

The following error messages are possible.

**Table 6-1. Error Messages for Output Modules with Error Diagnostics**

| Address | Type of Error |
|---|---|
| I X.0 | Short circuit on an output channel / fuse blown<br>or<br>no-load voltage |
| I X.1 | Defective module (output transistor shorted) |

X is the byte address (slot number) of the output module

Signal state “1” indicates an error is present. The PII is set to “0” for output modules without error diagnostics.

### 6.4.2 Digital Input/Output Module, 16 Inputs, 16 Outputs, 24 V DC for All CPUs Version 8MA02 and Higher and for CPU 102, Version 8MA01, Revision 5 and Higher

Plug the module only into slots 0 through 7.
This module occupies the same address space as an analog module. However, only the first two of the eight reserved bytes are used.

The address consists of byte address n or n+1 and channel number Y. "n" is the start address of a slot, the first of the reserved bytes (e.g., byte 64 for slot 0). "n+1" is therefore the second of the reserved bytes. The designations "n" and "n+1" are printed on the frontplate of the module.

The input and output information occupies the same addresses.

The channel number is defined by the connection of the actuators and sensors to the crimp connector. The channel numbers are printed on the frontplate.

**Table 6-2. Address Assignment**

| Slot Number | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|
| Address PII (IN) and PIQ (OUT) | Channel n.0 to n.7 | 64.0 to 64.7 | 72.0 to 72.7 | 80.0 to 80.7 | 88.0 to 88.7 | 96.0 to 96.7 | 104.0 to 104.7 | 112.0 to 112.7 | 120.0 to 120.7 |
| | Channel n+1.0 to n+1.7 | 65.0 to 65.7 | 73.0 to 73.7 | 81.0 to 81.7 | 89.0 to 89.7 | 97.0 to 97.7 | 105.0 to 105.7 | 113.0 to 113.7 | 121.0 to 121.7 |

**Examples:** Determining the Address

1) You plugged the module into slot 4 and connected an actuator at byte n, channel 4. The address is 96.4.

2) Address 113.3 indicates a sensor or an actuator is connected at byte n+1, channel 3. The module is plugged into slot 6.

### 6.4.3 Function Modules

Function modules have module-specific addressing. Some function modules are addressed like digital modules, and other function modules are addressed like analog modules. The addressing for each function module is explained in chapter 15.

## 6.5 The Structure of Process Image Input and Output Tables

Information about inputs is stored in the process image input table (PII). Information about outputs is stored in the process image output table (PIQ).

The PII and the PIQ each have an area of 128 bytes in the RAM memory.

The PII and the PIQ have identical structures. The PII and the PIQ can be divided into three address areas as shown in Table 6-3.

**Table 6-3. Structure of the PII and the PIQ**

| Byte Address in the PII and PIQ | Module | Slot Number |
|---|---|---|
| 0 to 31 | Digital modules | 0 to 31 |
| 32 to 63 | Unassigned address space | |
| 64 to 127 | Analog modules | 0 to 7 |

- The address space for bytes 0 through 31 is reserved for information from or to modules that are addressed like digital modules.
- The unassigned address space in bytes 32 to 63 can be used to store intermediate results.
- The address space in bytes 64 to 127 is reserved for information from or to modules that are addressed like analog modules.

Figure 6-7 shows a possible programmable controller configuration and storage of information in the process I/O images.

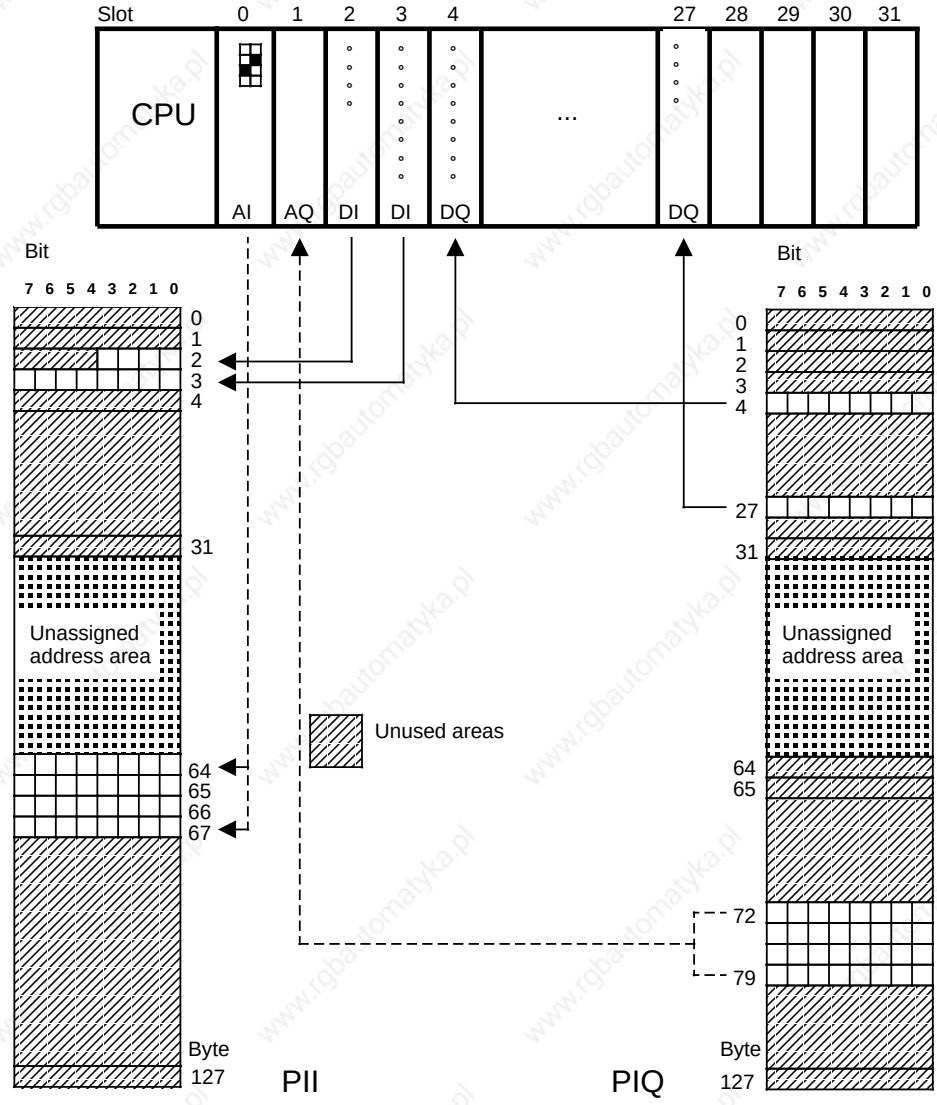



**Figure 6-7. Assignment of Process Images to the I/O Modules**

### 6.5.1 Accessing the Process Image Input Table (PII)

During a data cycle, data is read into the process image input table (PII) from input modules (see section 2.2.2 - Data Cycle). This data is available to the control program for evaluation in the next program processing cycle.

Access to the PII is expressed by the operand identifiers “I”, “IB”, or “IW” in a statement in the control program.
The letter “L” identifies the “Load” operation (see chapter 8). The letter “A” identifies the “AND logic” operation (see chapter 8).

- Bit-by-bit reading “I <bit address>”
  Example: Reading in the signal state of channel 2 of a 4-channel digital input module in slot 2

- Byte-by-byte reading “IB <byte address>”
  Example: Reading in the signal states of all channels of an 8-channel digital input module in slot 12

- Word-by-word reading “IW <word address>”
  Example: Reading in the analog value of channel 3 of a 4-channel analog input module in slot 4

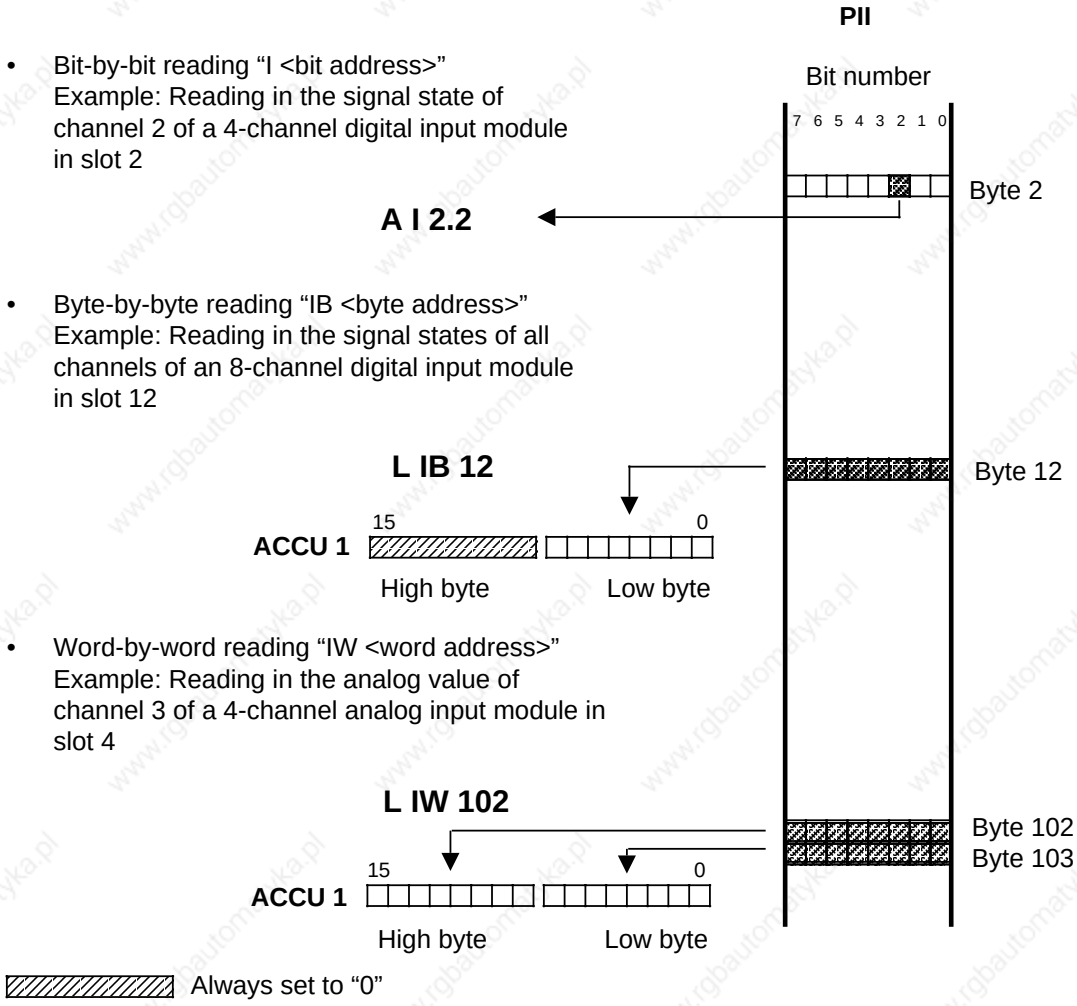
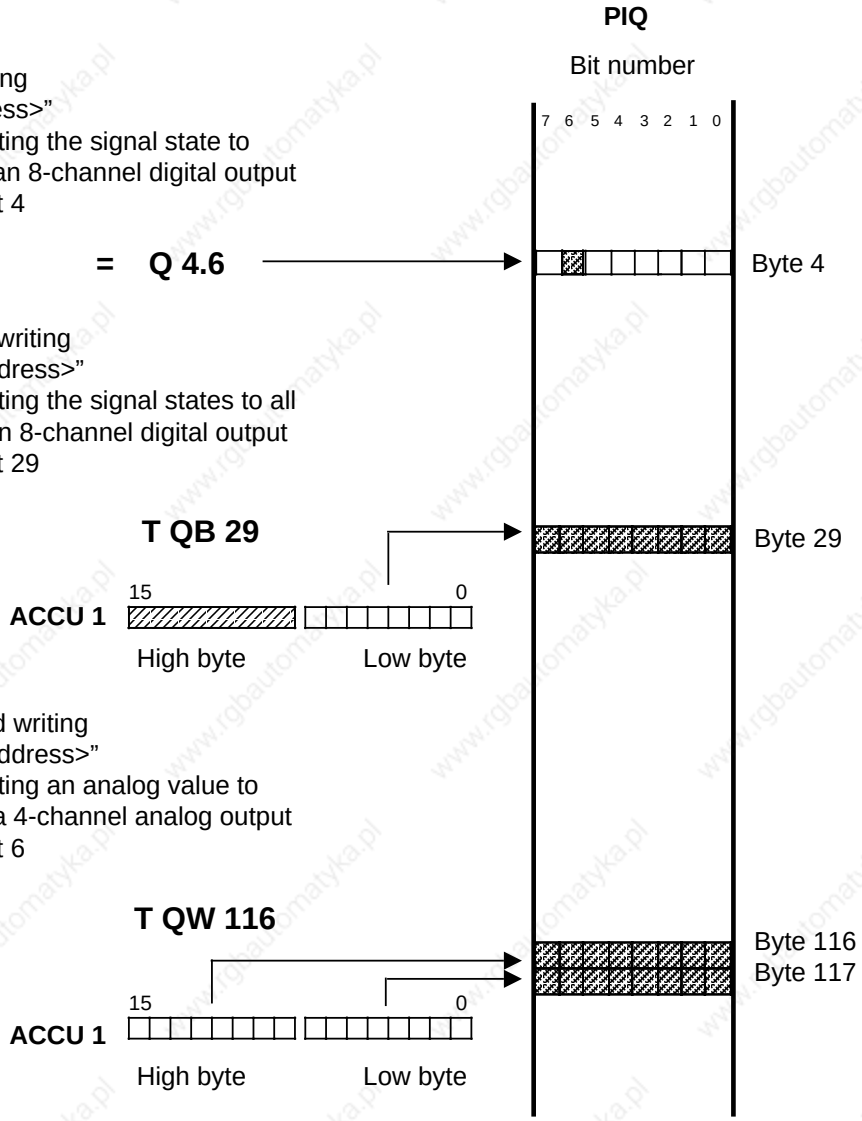



**Figure 6-8. Accesses to the PII**

### 6.5.2 Accessing the Process Image Output Table (PIQ)

During a program cycle, data coming from the control program to the output modules is written into the process image output table (PIQ). The data is transferred to the output modules in the following data cycle.

Access to the PIQ is expressed by the operand identifiers “Q”, “QB”, or “QW” in a statement in the control program.
The letter “T” identifies the “Transfer” operation (see Chapter 8). The “=” character assigns the result of a logic operation (RLO) to the operand that follows the character (see chapter 8).




- Bit-by-bit writing
  “Q <bit address>”
  Example: Writing the signal state to channel 6 of an 8-channel digital output module in slot 4
- Byte-by-byte writing
  “QB <byte address>”
  Example: Writing the signal states to all channels of an 8-channel digital output module in slot 29
- Word-by-word writing
  “QW <word address>”
  Example: Writing an analog value to channel 2 of a 4-channel analog output module in slot 6

**Figure 6-9. Accesses to the PIQ**

## 6.6 Interrupt Process Images and Time-Controlled Program Processing in OB13 for CPU 103, Version 8MA02 and Higher

In the event of a time-controlled or process interrupt, the CPU does not access the I/O modules directly. The CPU stores its information in interrupt process images.

- The interrupt process images are used only for time-controlled or interrupt-driven program processing.
- The interrupt process images and the “normal” process images have identical structures.
- The interrupt process input image (interrupt PII) and interrupt process output image (interrupt PIQ) take up an area of 128 bytes each in the RAM.

The interrupt PII and interrupt PIQ can be divided into three address areas as shown in Table 6-4.

**Table 6-4. Structure of the Interrupt PII and the Interrupt PIQ**

| Byte address in interrupt PII and interrupt PIQ | Module | Slot number |
|---|---|---|
| 0 to 31 | Digital modules | 0 to 31 |
| 32 to 63 | Unassigned address space | |
| 64 to 127 | Analog modules | 0 to 7 |

**Note**

The interrupt process images can be accessed by byte or word operations only.

### 6.6.1 Accessing the Interrupt PII

- The interrupt PII can only be accessed in connection with time-controlled or interrupt-driven program processing.
- Data from inputs is read into the interrupt PII only at the beginning of time-controlled program processing. This data is available only to the time-controlled program for evaluation.

### Time-Controlled Program Processing

Access to the interrupt PII is expressed by the “PB” or “PW” operand identifiers in a statement in the time-controlled program.
The letter “L” represents the “Load” operation (see chapter 8).

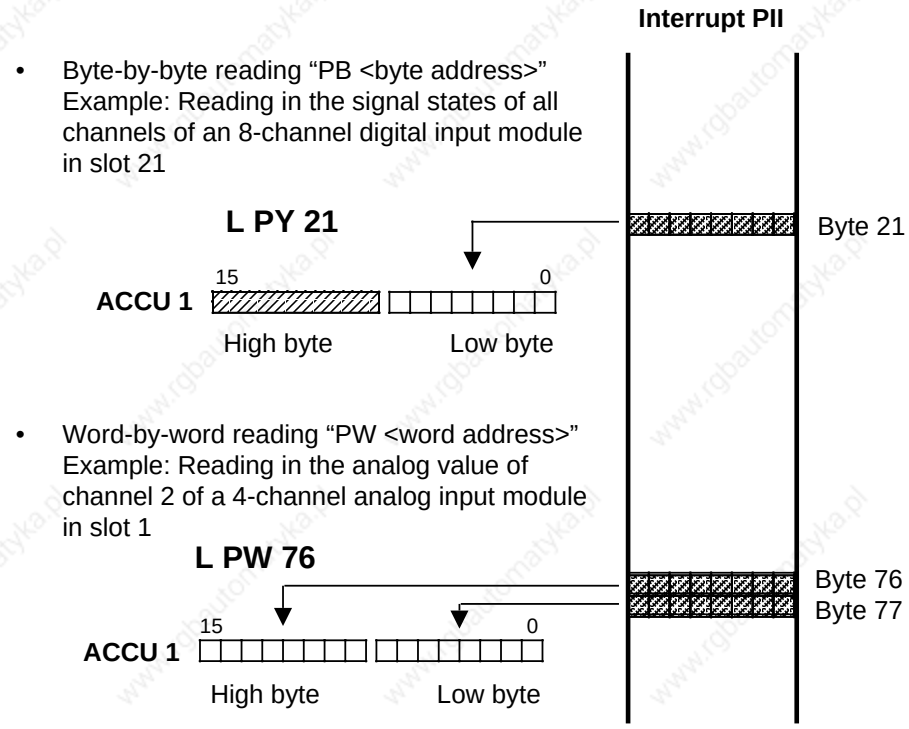



**Figure 6-10. Accesses to the Interrupt PII**

### Interrupt-Driven Program Processing

- When a process interrupt occurs, only the data of the interrupt inputs, slots 0 and 1, is read into the interrupt PII.
- Only this data of the interrupt PII is available to the interrrupt-driven program for evaluation.
- In a statement in the interrrupt-driven program, access to the interrupt PII is possible only with the following operands: PB0, PB1, and PW0.
- If other parameters are specified, the CPU goes into the STOP mode and the “NNN” error message is specified in the ISTACK. See section 5.2.

### 6.6.2 Accessing the Interrupt PIQ

When accessing the interrupt PIQ, the following rules apply.

- Data can be written to the interrupt PIQ only within time-controlled or interrupt-driven program processing.
- Data from a time-controlled or interrupt-driven program to external outputs is written during time-controlled or interrupt-driven program processing both to the “normal” PIQ and the interrupt PIQ.
- Data from the interrupt PIQ is read out to the outputs in the next interrupt output data cycle.
- The PIQ is copied to the interrupt PIQ after the OB1 program cycle.

**Note**

The interrupt output data cycle is executed only after the interrupt PIQ has been written to.

Access to the interrupt PIQ is expressed by the “PB” or “PW” operand identifiers in a statement in the time-controlled or interrrupt-driven program.
The letter “T” identifies the “Transfer” operation (see chapter 8).

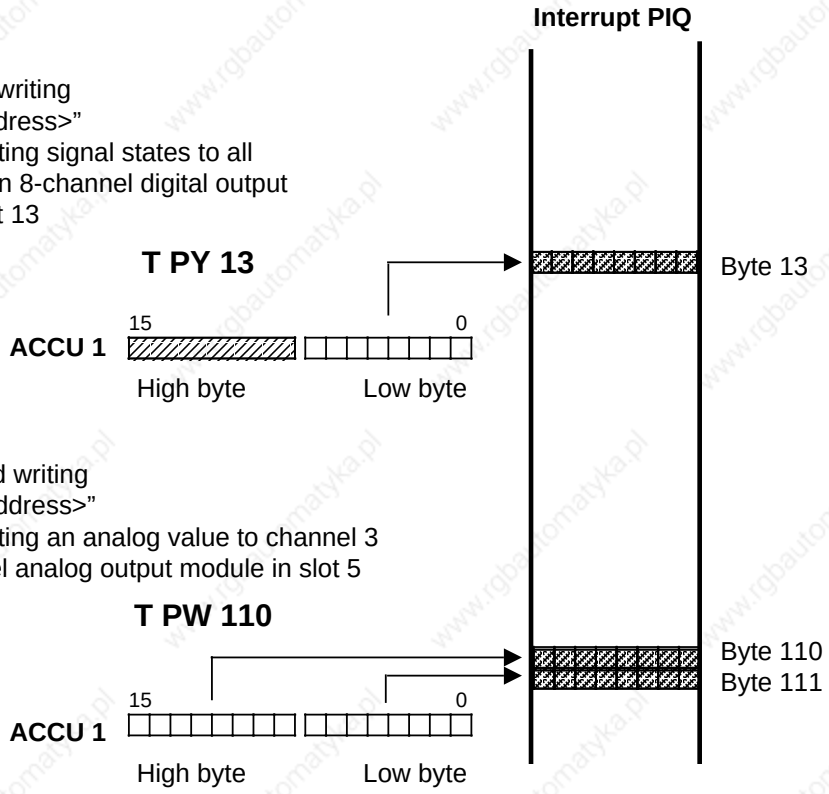



**Figure 6-11. Accesses to the Interrupt PIQ**

## 6.7 RAM Address Assignments

The following table gives an overview of the major addresses in the RAM of the three CPUs (in hexadecimal code).

**Table 6-5. Important Addresses in the RAM**

| CPU | 100 | 102* | 103 |
|---|---|---|---|
| Program memory | EE00 to FFFF | D000 to DFFF | 8000 to CFFF |
| Memory submodule | C000 to DFFF | 4000 to 5FFF | 0000 to 7FFF |
| PII, digital | E400 to E41F | EF00 to EF1F | EF00 to EF1F |
| PII, analog | E440 to E47F | EF40 to EF7F | EF40 to EF7F |
| PIQ, digital | E480 to E49F | EF80 to EF9F | EF80 to EF9F |
| PIQ, analog | E4C0 to E4FF | EFC0 to EFFF | EFC0 to EFFF |
| Timers | E280 to E29F | EC00 to EC39 | EC00 to ECFF |
| Retentive counters | E2A0 to E2AF | ED00 to ED0F | ED00 to ED0F |
| Non-retentive counters | E2B0 to E2BF | ED10 to ED3F | ED10 to ED3F |
| Retentive flags | E300 to E33F | EE00 to EE3F | EE00 to EE3F |
| Non-retentive flags | E340 to E37F | EE40 to EE7F | EE40 to EE7F |
| **Module address list** | | | |
| OB | E080 to E0FF | FC80 to FCFF | DC00 to DDFF |
| FB | E100 to E17F | FD00 to FEFF | DE00 to DFFF |
| PB | E180 to E1FF | FF00 to FF7F | E000 to E1FF |
| SB | ---- | ---- | E200 to E3FF |
| DB | E200 to E27F | FF80 to FFFF | E400 to E5FF |
| System data | EA00 to EBFF | EA00 to EBFF | EA00 to EBFF |

* Program memory; block address list only in TEST mode.

The following table gives an overview of the most important system data in the system data area.

**Table 6-6. System Data Area Assignment**

| System data word | Contents | Chapter/ Section Reference |
|---|---|---|
| 5 to 7 | ISTACK (Interrupt STACK) | 5.2 |
| 8 to 12 | Integral real-time clock | 12 |
| 33 | First free program memory address | |
| 35 | Program memory starting address | |
| 37 | Program memory end address | |
| 40 to 45 | CPU version, software release | |
| 57 to 63 | SINEC L1 | 13 |
| 96 | Scan monitoring time (value · 10 ms) | |
| 97 | Calling interval for OB 13 for time-controlled program processing (value · 10 ms) | 7.4.4 |
| 128 to 159 | BSTACK (Block STACK) | 5.3.2 |
| 203 to 214 | ISTACK (Interrupt STACK) | 5.2 |

## 7 Introduction to STEP 5


| | | |
|---|---|---|
| 7.1 | Writing a Program | 7 - 1 |
| 7.1.1 | Methods of Representation | 7 - 1 |
| 7.1.2 | Operand Areas | 7 - 3 |
| 7.1.3 | Circuit Diagram Conversion | 7 - 3 |
| 7.2 | Program Structure | 7 - 4 |
| 7.2.1 | Linear Programming | 7 - 4 |
| 7.2.2 | Structured Programming | 7 - 5 |
| 7.3 | Block Types | 7 - 7 |
| 7.3.1 | Organization Blocks | 7 - 9 |
| 7.3.2 | Program Blocks | 7 - 11 |
| 7.3.3 | Sequence Blocks, for CPU 103 and Higher | 7 - 11 |
| 7.3.4 | Function Blocks | 7 - 11 |
| 7.3.5 | Data Blocks | 7 - 16 |
| 7.4 | Program Processing | 7 - 18 |
| 7.4.1 | Program Processing with CPU 102 | 7 - 19 |
| 7.4.2 | START-UP Program Processing | 7 - 24 |
| 7.4.3 | Cyclic Program Processing | 7 - 26 |
| 7.4.4 | Time-Controlled Program Processing, for CPU 103 Version 8MA02 and Higher | 7 - 28 |
| 7.4.5 | Interrupt-Driven Program Processing, for CPU 103 Version 8MA02 and Higher | 7 - 29 |
| 7.5 | Processing Blocks | 7 - 30 |
| 7.5.1 | Changing Programs | 7 - 30 |
| 7.5.2 | Changing Blocks | 7 - 30 |
| 7.5.3 | Compressing the Program Memory | 7 - 30 |
| 7.6 | Number Representation | 7 - 31 |

## Figures


7-1 Compatibility of STEP 5 Methods of Representation . . . . . 7 - 2
7-2 Nesting Depth of Programmed Organization Blocks . . . . . 7 - 6
7-3 Structure of a Block Header . . . . . 7 - 8
7-4 Example of Organization Block Use . . . . . 7 - 10
7-5 Programming a Function Block Parameter, for CPU 103 and Higher . . . . . 7 - 13
7-6 Programming a Function Block . . . . . 7 - 16
7-7 Example of Data Block Contents . . . . . 7 - 17
7-8 Validity Areas of Data Blocks . . . . . 7 - 17
7-9 Programm Scanning with CPU 102 . . . . . 7 - 19
7-10 Mode Change for CPU 102 . . . . . 7 - 21
7-11 Display of the Processing Mode in the ISTACK . . . . . 7 - 22
7-12 Setting the Start-Up Procedure . . . . . 7 - 24
7-13 Cyclic Program Processing . . . . . 7 - 26
7-14 Calculating the Response Time . . . . . 7 - 27
7-15 Compressing the Program Memory . . . . . 7 - 30
7-16 Bit Assignment of a 16-Bit Fixed-Point Binary Number . . . . . 7 - 31
7-17 BCD and Decimal Formats . . . . . 7 - 32


## Tables


7-1 Comparison of Operation Types . . . . . 7 - 2
7-2 Comparison of Block Types . . . . . 7 - 7
7-3 Overview of Organization Blocks . . . . . 7 - 9
7-4 Block Parameter Types and Data Types with Permissible Actual Parameters, for CPU 103 and Higher . . . . . 7 - 14
7-5 Programming Possibilities . . . . . 7 - 18
7-6 Comparison of Number Formats . . . . . 7 - 32

# 7 Introduction to STEP 5

This chapter explains how to program the S5-100U. It describes how to write a program, how the program is structured, the types of blocks the program uses, and the number representation of the STEP 5 programming language.

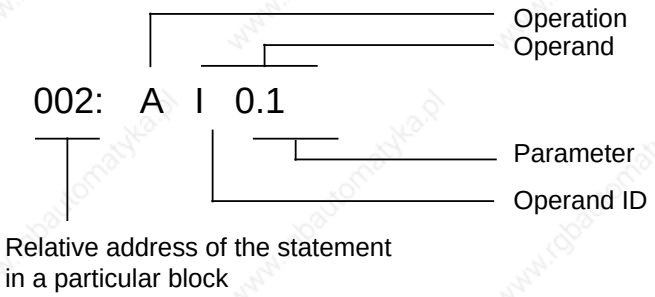
## 7.1 Writing a Program

A control program specifies a series of operations that tell the programmable controller how it has to control a system. For example, a control program might be the series of operations that tell the S5-100U how to use open-loop control or closed-loop control for a specific system. You must write the program in a special programming language and according to specific rules so that the programmable controller can understand it. The programming language that has been developed for the SIMATIC S5 family is called STEP 5.

### 7.1.1 Methods of Representation

The following methods of representation are possible with the STEP 5 programming language.

- **Statement List (STL)**
  STL represents the program as a sequence of operation mnemonics. A statement has the following format:




  The operation tells the programmable controller what to do with the operand. The parameter indicates the operand address.

- **Control System Flowchart (CSF)**
  CSF represents logic operations with graphics symbols.
- **Ladder Diagram (LAD)**
  LAD graphically represents control functions with circuit diagram symbols.
- **GRAPH 5, for CPU 103 and higher**
  GRAPH 5 describes the structure of sequence control systems.

You cannot use CSF, LAD, or GRAPH 5 with the PG 605 programmers.

Each method of representation has its own special characteristics. A program block that has been programmed in STL cannot necessarily be output in CSF or LAD. The three methods of graphic representation are not compatible. However, programs in CSF or LAD can always be converted to STL. Figure 7-1 illustrates these points in a diagram.

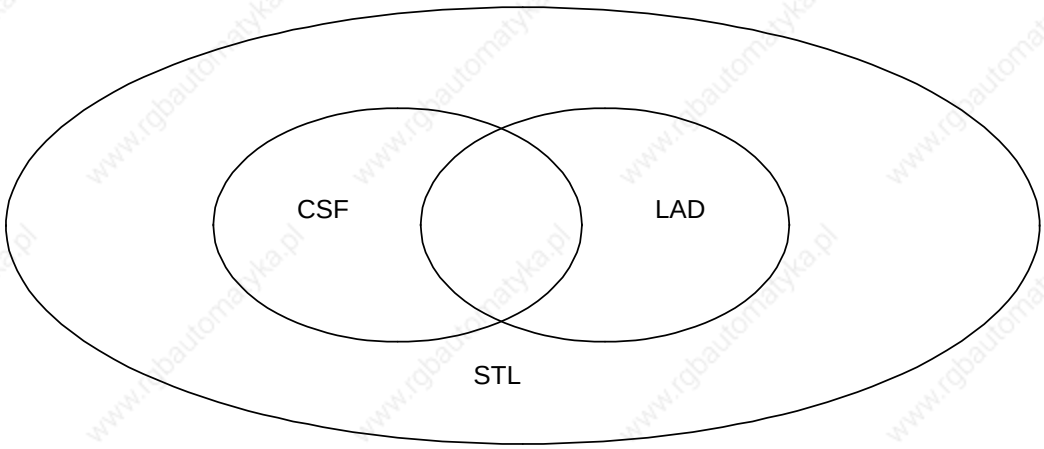



**Figure 7-1. Compatibility of STEP 5 Methods of Representation**

The STEP 5 programming language has the following three operation types:

- Basic
- Supplementary
- System

Table 7-1 provides further information about these operations.

**Table 7-1. Comparison of Operation Types**

| STEP 5 PROGRAMMING LANGUAGE | | | |
|---|---|---|---|
| | **Basic Operations** | **Supplementary Operations** | **System Operations** |
| Application | In all blocks | Only in function blocks | Only in function blocks |
| Methods of representation | STL, CSF, LAD | STL | STL |
| Special features | | | For users with good system knowledge |

Refer to Chapter 8 for a description of all operations and for programming examples.

### 7.1.2 Operand Areas

The STEP 5 programming language has the following operand areas:

| | | |
|---|---|---|
| **I** | (inputs) | Interfaces from the process to the programmable controller |
| **Q** | (outputs) | Interfaces from the programmable controller to the process |
| **F** | (flags) | Memory for intermediate results of binary operations |
| **D** | (data) | Memory for intermediate results of digital operations |
| **T** | (timers) | Memory for implementing timers |
| **C** | (counters) | Memory for implementing counters |
| **P** | (peripherals) | Interfaces from the process to the programmable controller |
| **K** | (constants) | Defined numeric values |
| **OB, PB, SB FB, DB** | (blocks) | Program structuring aids |

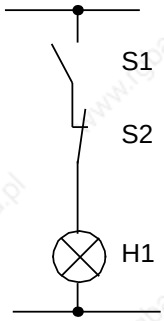
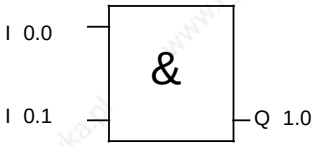
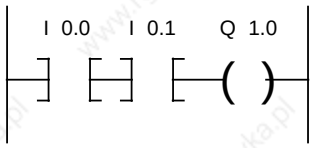
Refer to Appendix A for a listing of all operations and operands.

### 7.1.3 Circuit Diagram Conversion

If your automation task is in the form of a circuit diagram, you must convert it to STL, CSF, or LAD.

**Example:** **Hard-Wired Control**
A signal lamp (H1) is supposed to light up when a normally open contact (S1) is activated and a normally closed contact (S2) is not activated.

**Programmable Control**
The signal lamp is connected to an output (i.e., Q 1.0). The signal voltages of the two contacts are connected to two programmable controller inputs (i.e., I 0.0 and I 0.1). The S5-100U scans to see if the signal voltages are present (signal state “1” at the activated normally open contact or non-activated normally closed contact). Both signal states are combined through logic AND. The result of logic operation (RLO) is assigned to output Q 1.0 (the lamp lights).

| Circuit Diagram | STL | CSF | LAD |
|---|---|---|---|
|   | A I 0.0<br>A I 0.1<br>= Q 1.0 |   |   |

## 7.2 Program Structure

An S5-100U program can be one of the two following types:

- Linear
- Structured

Sections 7.2.1 and 7.2.2 describe these program types.

### 7.2.1 Linear Programming

Programming individual operations in one section (block) is sufficient for handling simple automation jobs. For the S5-100U, this is organization block 1 (see section 7.3.1). The S5-100U scans this block cyclically. After the S5-100U scans the last statement, it goes back to the first statement and begins scanning again. Please note the following rules:

- When OB1 is called, five words are assigned to the block header in the program memory (see section 7.3).
- Normally, a statement takes up one word in the program memory.
  Two-word statements also exist (e.g., with the operation “Load a constant”). Count these statements twice when calculating the program length.
- Like all blocks, OB1 must be terminated by a Block End statement (BE).

### 7.2.2 Structured Programming

To solve complex tasks, it is advisable to divide a program into individual, self-contained program parts (blocks). This procedure has the following advantages:

- Simple and clear programming, even for large programs
- Program parts can be standardized
- Easy alterations
- Simple program test
- Simple start-ups
- Subroutine techniques (block call from different locations)

The STEP 5 programming language has the following five block types:

- **Organization Block (OB)**
  Organization blocks manage the control program.
- **Program Block (PB)**
  Program blocks arrange the control program according to functional or technical aspects.
- **Sequence Block (SB)**
  Sequence blocks are special blocks that program sequence controls. They are handled like program blocks. (This is available for CPU 103 and higher.)
- **Function Block (FB)**
  Function blocks are special blocks for programming frequently recurring or especially complex program parts (e.g., reporting and arithmetic functions). You can assign parameters to them (available for CPU 103 and higher). They have an extended set of operations (e.g., jump operations within a block).
- **Data Block (DB)**
  Data blocks store data needed to process a control program. Actual values, limiting values, and texts are examples of data.

The program uses block calls to exit one block and jump to another. You can therefore nest program, function, and sequence blocks randomly up to 16 levels (see section 7.3). Nesting can be up to 32 levels for CPU 103 version 8MA03.

> **Note**
>
> When calculating the nesting depth, note that the system program in the programmable controller can call an organization block automatically under certain circumstances (e.g., OB2).

The total nesting depth is the sum of the nesting depths of call programmed organization blocks. If nesting goes beyond 16 levels (32 levels for CPU 103 version 8MA03), the CPU goes into the STOP mode with the error message “STUEB,” block stack overflow (see section 5.2). Figure 7-2 illustrates the nesting principle.

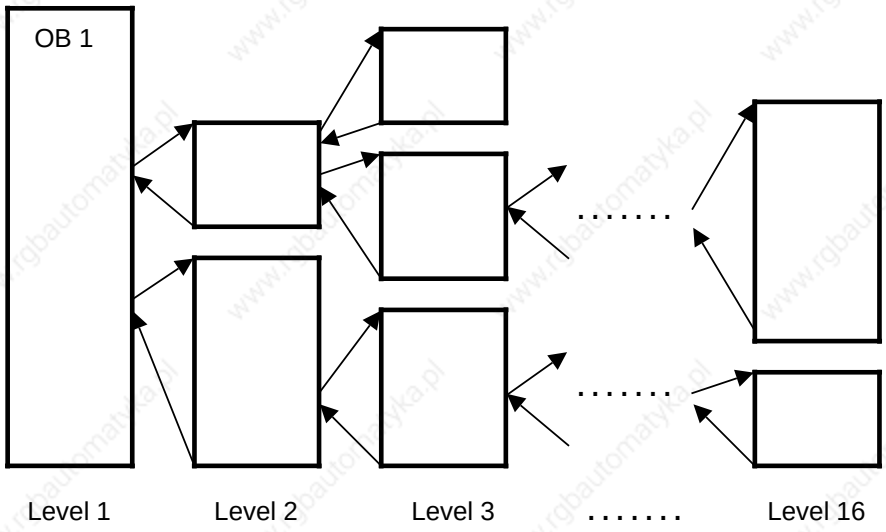



**Figure 7-2. Nesting Depth of Programmed Organization Blocks**

## 7.3 Block Types

The following table lists the most important characteristics of the individual block types:

**Table 7-2. Comparison of Block Types**

| | OB[1] | PB | SB | FB[2] | DB[3] |
|---|---|---|---|---|---|
| Number CPU 100 | 64 OB0 to OB63 | 64 PB0 to PB63 | ——— | 64 FB0 to FB63 | 62 DB2 to DB63 |
| Number CPU 102 | 64 OB0 to OB63 | 64 PB0 to PB63 | ——— | 64 FB0 to FB63 | 62 DB2 to DB63 |
| Number CPU 103 | 256 OB0 to OB255 | 256 PB0 to PB255 | 256 SB0 to SB255 | 256[2] FB0 to FB255 | 254 DB2 to DB255 |
| Length (max.) CPU 100 | 4 Kbytes | 4 Kbytes | ——— | 4 Kbytes | 256 data words |
| Length (max.) CPU 102 | 4 Kbytes | 4 Kbytes | ——— | 4 Kbytes | 256 data words |
| Length (max.) CPU 103 | 8 Kbytes | 8 Kbytes | 8 Kbytes | 8 Kbytes | 8 Kbytes |
| Operations set (contents) | Basic operations | Basic operations | Basic operations | Basic, supple-mentary, system operations | Bit patterns, numbers, texts |
| Representa-tion methods | STL, CSF, LAD | STL, CSF, LAD | STL, CSF, LAD | STL | |
| Block header length | 5 words | 5 words | 5 words | 5 words | 5 words |

1 The operating system calls up particular OBs automatically (see section 7.3.1 and 9.3).
2 Function blocks are already integrated into the operating system (see section 9.2).
3 Data blocks DB0 and DB1 are reserved.

### Block Structure

Each block consists of the following parts:

- The block header that specifies the block type, number, and length
  - Generated by the programmer when it transforms the block
- The block body that has the STEP 5 program or data

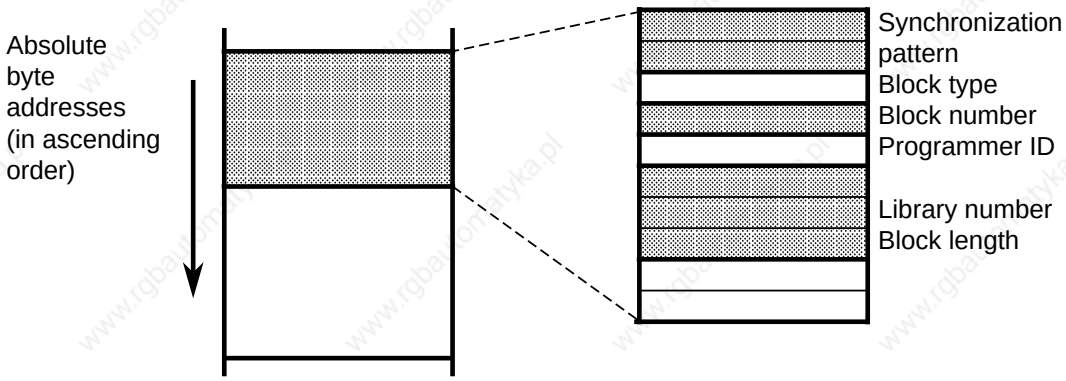



**Figure 7-3. Structure of a Block Header**

### Programming

Program your blocks as follows (does not apply to data blocks):

1. Specify the block type (e.g., PB).
2. Specify the block number (e.g., 27).
3. Enter the control program statements.
4. Terminate the block with the “BE” statement.

### 7.3.1 Organization Blocks

Organization blocks (OB) form the interface between the operating system and the control program. Organization blocks are handled in one of the following three ways:

- Organization block OB1 is called cyclically by the operating system.
- Some organization blocks are event-driven or time-controlled. They can be called in response to events or at certain times:
  - By a switch from STOP to RUN (OB21)
  - By a switch from Power OFF to Power ON (OB22 (see Table 7-3))
  - By interrupts (OB2 and OB13)
- Some other organization blocks represent operating functions (similar to the the integral function blocks). They can be called by the control program (for CPU 103 and higher; see section 9.3).

Table 7-3 provides an overview of organization blocks.

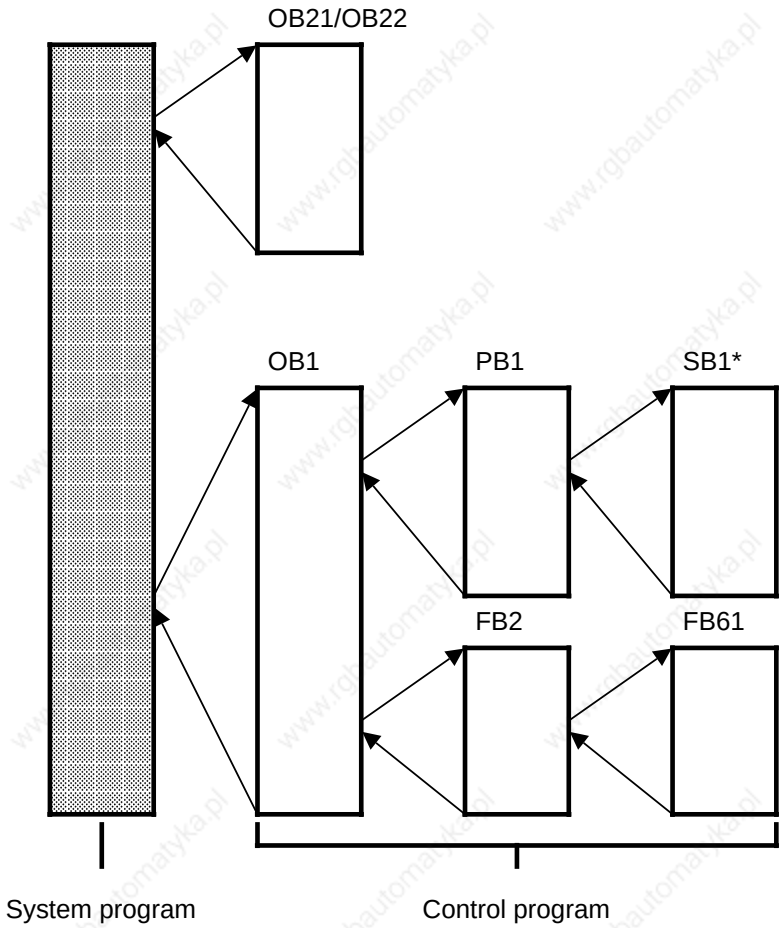
**Table 7-3. Overview of Organization Blocks**

| OB No. | Function | OB integrated in | | |
|---|---|---|---|---|
| | | CPU 100 | CPU 102 | CPU 103 |
| **You must program the OB. The operating system calls up the OB.** | | | | |
| OB1 | Cyclic program processing | ▒ | ▒ | ▒ |
| **Interrupt-driven program processing** | | | | |
| OB2 | Interrupt-driven program processing | | | ▒ |
| OB13 | Time-controlled program processing | | | ▒ |
| **Handling start-up procedures** | | | | |
| OB21 | When starting manually (STOP to RUN) | ▒ | ▒ | ▒ |
| OB22 | When power returns | ▒ | ▒ | ▒ |
| **Handling programming errors and device errors** | | | | |
| OB34 | Battery failure | ▒ | ▒ | ▒ |
| **The OB is already programmed. You must call up the OB.** | | | | |
| OB31 | Scan time triggering (resets scan time monitor) | | | ▒ |
| OB251 | PID control algorithm | | | ▒ |

▒ OB is ready or is supported by the operating system

You can program all organization blocks using parameters from the permissible range. CPU 100 and CPU 102 use organization blocks OB0 to OB63. CPU 103 uses OB0 to OB255. However, you must call the organization blocks from the control program.

Figure 7-4 shows how to set up a structured control program. It also illustrates the significance of organization blocks.




* For CPU 103 and higher

**Figure 7-4. Example of Organization Block Use**

### 7.3.2 Program Blocks

Self-contained program parts are programmed in program blocks (PB).

Special feature: Control functions can be represented graphically in program blocks.

**Call**

Block calls JU and JC activate program blocks. You can program these operations in all block types except data blocks. Block call and block end cause the RLO to be reloaded. However, the RLO can be included in the “new” block and be evaluated there.

### 7.3.3 Sequence Blocks, for CPU 103 and Higher

Sequence blocks (SB) are special program blocks that process sequence controls. They are treated like program blocks.

### 7.3.4 Function Blocks

Frequently recurring or complex control functions are programmed in function blocks (FB).

Function blocks have the following special features.

- FBs can be assigned parameters (for CPU 103 and higher).
  - Actual parameters can be assigned when the block is called (for CPU 103 and higher).
- FBs have an extended set of operations not available to other blocks.
- The FB program can be written and documented in STL only.

If you are using CPU 102 version 8MA02 or higher, you have the following types of function blocks available:

- FBs that you can program
- FBs that are integrated in the operating system (see section 9.2)
- FBs that are available as software packages (standard function blocks, see Catalog ST 57)

**Block Header**

Besides the block header, function blocks have organizational information that other blocks do not have.

A function block's memory requirements consist of the following:

- Block header (five words) as for other blocks
- Block name (five words)
- Block parameter for parameter assignment (three words per parameter)

**Creating a Function Block, for CPU 103 and Higher**

In contrast to other blocks, parameters can be assigned to FBs.
To assign parameters, you must program the following block parameter information.

- **Block Parameter Name (formal operand)**
  Each block parameter as a formal operand is given a designation (DES). Under this designation it is replaced by an actual parameter when the function block is called.
  The name can be up to four characters long and must begin with an alpha character. You can program up to 40 block parameters per function block.
- **Block Parameter Type**
  You can enter the following parameter types:
  - I input parameters
  - Q output parameters
  - D data
  - B blocks
  - T timers
  - C counters

  In graphic representation, output parameters appear to the right of the function symbol. Other parameters appear to the left.
- **Block Parameter Data Type**
  You can specify the following data types:
  - BI for operands with a bit address
  - BY for operands with a byte address
  - W for operands with a word address
  - K for constants

When assigning parameters, enter all block parameter specifications.

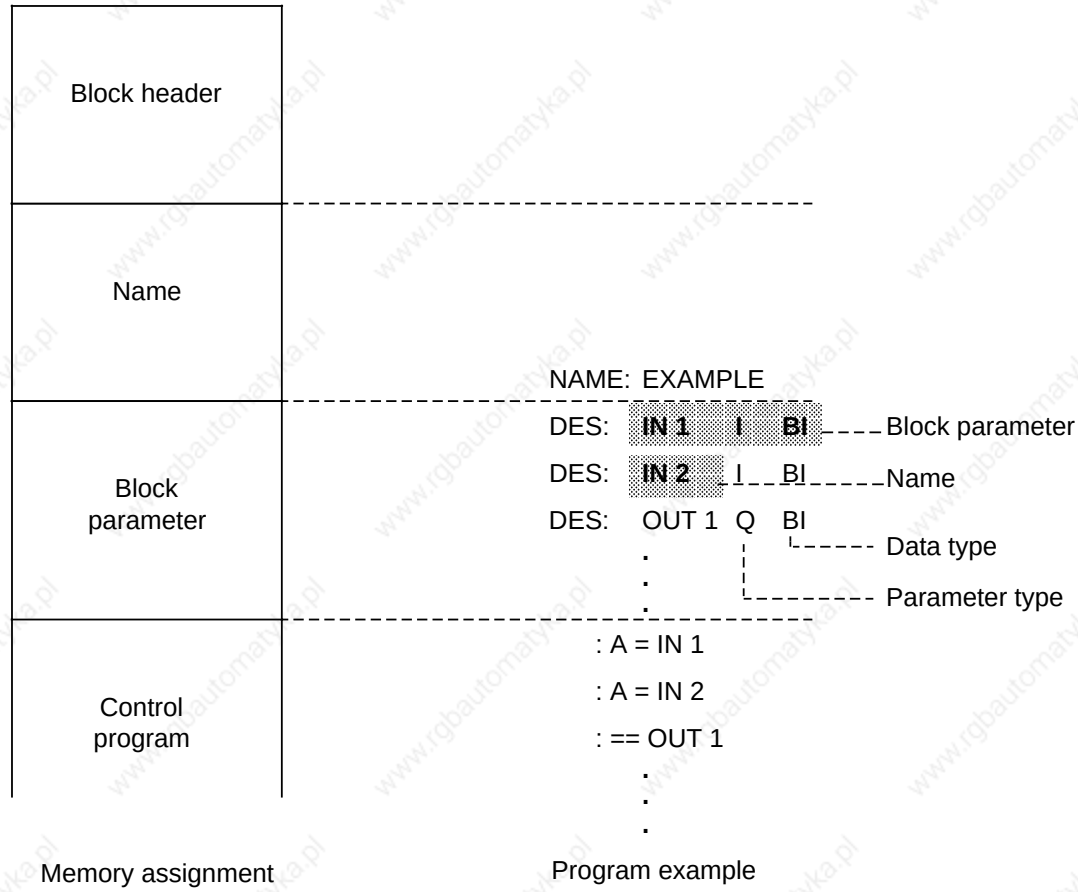



**Figure 7-5. Programming a Function Block Parameter, for CPU 103 and Higher**

**Table 7-4. Block Parameter Types and Data Types with Permissible Actual Parameters, for CPU 103 and Higher**

| Parameter Type | Data Type | Permissible Actual Parameters |
|---|---|---|
| I, Q | BI for an operand with bit address | I x.y Inputs<br>Q x.y Outputs<br>F x.y Flags |
| | BY for an operand with byte address | IB x Input bytes<br>QB x Output bytes<br>FY x Flag bytes<br>DL x Data bytes left<br>DR x Data bytes right<br>PY x Peripheral bytes* |
| | W for an operand with word address | IW x Input words<br>QW x Output words<br>FW x Flag words<br>DW x Data words<br>PW x Peripheral words* |
| D | KM for a binary pattern (16 digits)<br>KY for two absolute numbers, one byte each, each in the range from 0 to 255<br>KH for a hexadecimal pattern (maximum 4 digits)<br>KS for a character (maximum 2 alphanumeric characters)<br>KT for a time (BCD-coded time) with time base 1.0 to 999.3<br>KC for a count (BCD-coded) 0 to 999<br>KF for a fixed-point number in the range from -32768 to +32767 | Constants |
| B | Type designation not permitted | DBx Data blocks. The C DBx operation is executed.<br>OBx Organization blocks are called unconditionally (JU ... x).<br>FBx Function blocks (permissible without parameters only) are called unconditionally (JU..x).<br>PBx Program blocks are called unconditionally (JU..x).<br>SBx Sequence blocks are called unconditionally (JU..x). |
| T | Type designation not permitted | T Timer. The time should be assigned parameters as data or be programmed as a constant in the function block. |
| C | Type designation not permitted | C Counter. The count should be assigned parameters as data or be programmed as a constant in the function block. |

* Not permitted for integral FBs

### Calling a Function Block

Like other blocks, function blocks are stored under a specific number in the program memory (e.g., FB47). The numbers 240 to 255 are reserved for the integral function blocks (in CPU 103 version 8MA02 and higher).
You can program function block calls in all blocks except data blocks.

A function block call consists of the following parts:

- Call statement
  - JU FBx unconditional call (***J***ump ***U***nconditional)
  - JC FBx call if RLO = 1 (***J***ump ***C***onditional)
- Parameter list (only if block parameters were defined in the FB)

Function blocks can be called only if they have been programmed. When a function block call is being programmed, the programmer requests the parameter list for the FB automatically if block parameters have been defined in the FB.

**Setting Parameters for a Function Block**

The program in the function block specifies how the formal operands (parameters defined as “DES”) are to be processed.

As soon as you have programmed a call statement (for example JU FB2), the programmer displays the **parameter list**. The parameter list consists of the names of the parameters. Each parameter name is followed by a colon (:). You must assign actual operands to the parameters. The actual operands replace the formal operands defined in the FB when the FB is called, so that the FB operates with the actual operands.
A parameter list has a maximum of 40 parameters.

Example: The name (DES) of a parameter is IN1, the parameter type is I (as in input), the data type is BI (as in bit). The formal operand for the FB has the following structure:

DES: IN1 I BI

Specify in the parameter list of the calling block which actual operand is to replace the formal operand in the FB call. In our example it is : I 1.0.

Enter in the parameter list:

IN1: I 1.0

When the FB is called, it replaces the formal operand “IN1” with the actual operand “I 1.0”.

Figure 7-6 provides you with a detailed example of how to set parameters for a function block.

The FB call takes up two words in the internal program memory. Each parameter takes up an additional memory word.

You can find the memory requirements for standard function blocks and the run times in the specifications in Catalog ST 57.

The name of the function block is stored in the function block. The designations (DES) of the function block inputs and outputs that appear on the programmer during programming are also stored in the function block. Before you begin programming on the programmer, you must choose one of the following two options:

- Transfer all necessary function blocks to the program diskette (for off-line programming)
- Input all necessary function blocks directly into the program memory of the programmable controller

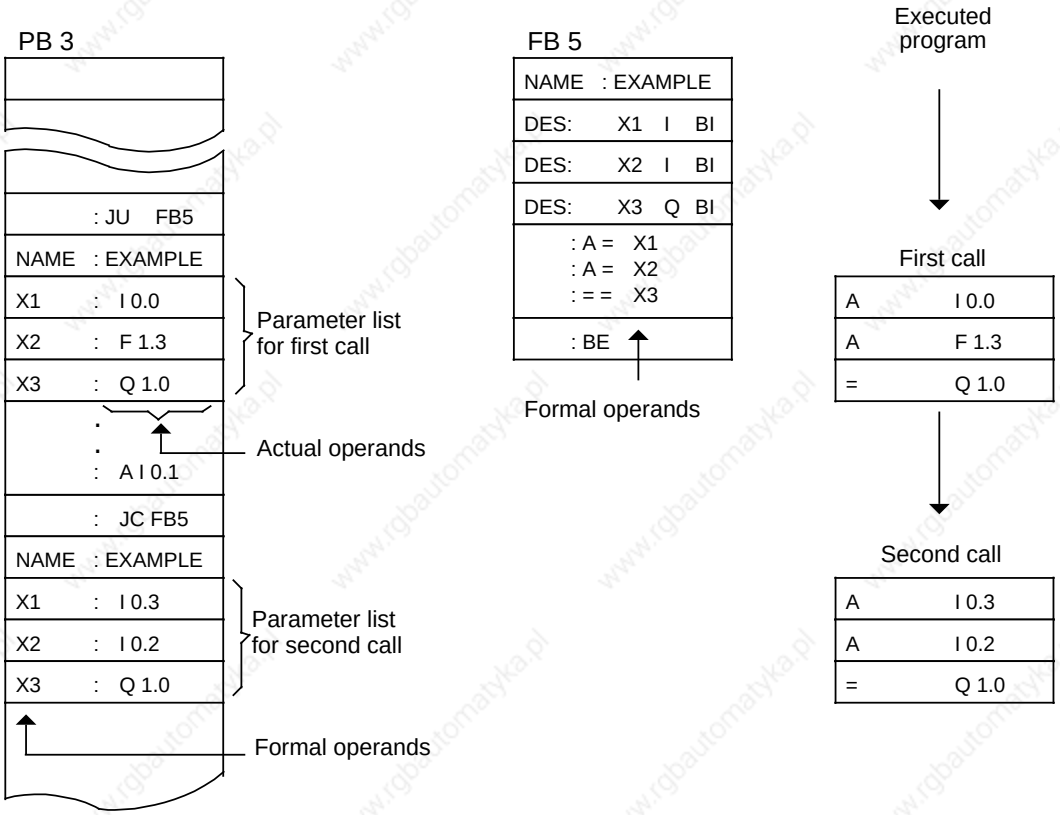



**Figure 7-6. Programming a Function Block**

### 7.3.5 Data Blocks

Data blocks (DB) store data to be processed in a program.

The following data types are permissible:

- Bit pattern (representation of controlled system states)
- Hexadecimal, binary or decimal numbers (times, results of arithmetic operations)
- Alphanumeric characters (ASCII message texts)

### Programming Data Blocks

Begin programming a data block by specifying a block number between 2 and 63 for CPU 100 or CPU 102, and between 2 and 255 for CPU 103. DB0 is reserved for the operating system, DB1 for setting parameters for internal functions (see section 9.1). Data is stored in this block in words.

If the information takes up less than 16 bits, the high-order bits are padded with zeros. Data input begins at data word 0 and continues in ascending order. A data block can hold up to 256 data words. You can call up or change the data word contents with load or transfer operations.

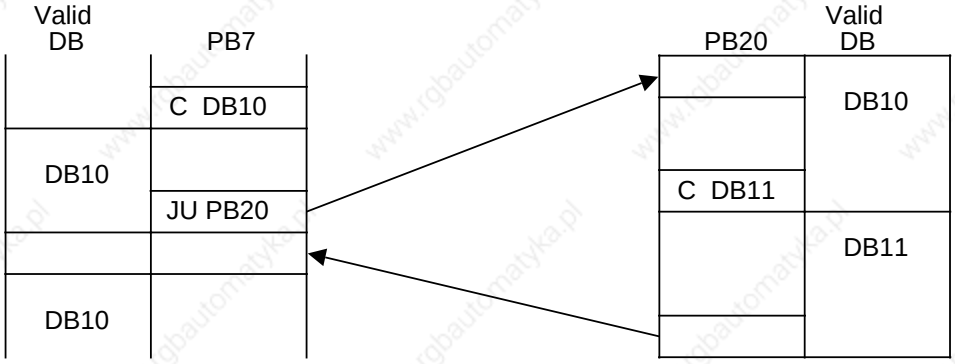
| Input | | | | | Stored Values | |
|---|---|---|---|---|---|---|
| 0000 | : | KH | = | A13C | DW0 | A13C |
| 0001 | : | KT | = | 100.2 | DW1 | 2100 |
| 0003 | : | KF | = | +21874 | DW2 | 5572 |

**Figure 7-7. Example of Data Block Contents**

You can also create or delete data blocks in the control program (see section 8.1.8).

### Program Processing with Data Blocks

- A data block must be called in the program with the C DBx operation (x = DB number) before it can be accessed.
- Within a block, a data block remains valid until another data block is called.
- When the program jumps back into the higher-level block, the data block that was valid before the block call is again valid.
- After OB1, 2, 13, 21, 22 have been called **by the operating system**, no DB is valid.




When PB20 is called, the valid data area is entered into memory.
When the program jumps back, this area is reopened.

**Figure 7-8. Validity Areas of Data Blocks**

### The Function of DB1

DB1 is used for special functions. DB1 is already integrated into CPU 103 version 8MA03 and higher and contains (default) values that you can either accept or change (see section 9.1). DB1 is evaluated once during start-up: either after Power ON or after a transition from STOP to RUN.

## 7.4 Program Processing

Some of the organization blocks (OBs) are responsible for structuring and managing the control program.

These OBS can be grouped according to the following assignments:

- OBs for START-UP program processing
- One OB for cyclic program processing
- OBs for time-controlled program processing
- OBs for (process) interrupt-driven program processing

The S5-100U has additional OBs whose functions are similar to those of integral function blocks (e.g., PID control algorithm). These OBs are described in chapter 9.

Section 7.3.1 summarizes all of the OBs.

**Comparing Programming Possibilities for CPU 100, CPU 102, and CPU 103**

**Table 7-5. Programming Possibilities**

| CPU | CPU 100 | CPU 102 | CPU 103 |
|---|---|---|---|
| Cyclical | Yes | Yes | Yes |
| Interrupt-driven | No | No | Yes<br>(for 8MA02 and higher) |
| Time-controlled | No | No | Yes<br>(for 8MA02 and higher) |
| Integral FBs | No | Yes<br>(for 8MA02 and higher) | Yes |
| Graph 5 | No | No | Yes |
| Programmable FBs | No | No | Yes |

Beginning with section 7.4.2, you learn which special organization blocks each of the CPUs has available to perform the programming tasks described in Table 7-5. You also learn which precautions you need to take when you program.

### 7.4.1 Program Processing with CPU 102

You can process the program in the following two modes:

- Normal mode
- Test mode

Program processing is faster in the normal mode, but you can not use the STATUS test function. Transferring from one mode to the other is called a mode change.

**Test Mode:**

Scanning the STEP 5 program

**Normal Mode**:

The control program you have written in STEP 5 is not processed directly. What is processed is a translated or runtime-optimized form of the program generated by the programmable controller.

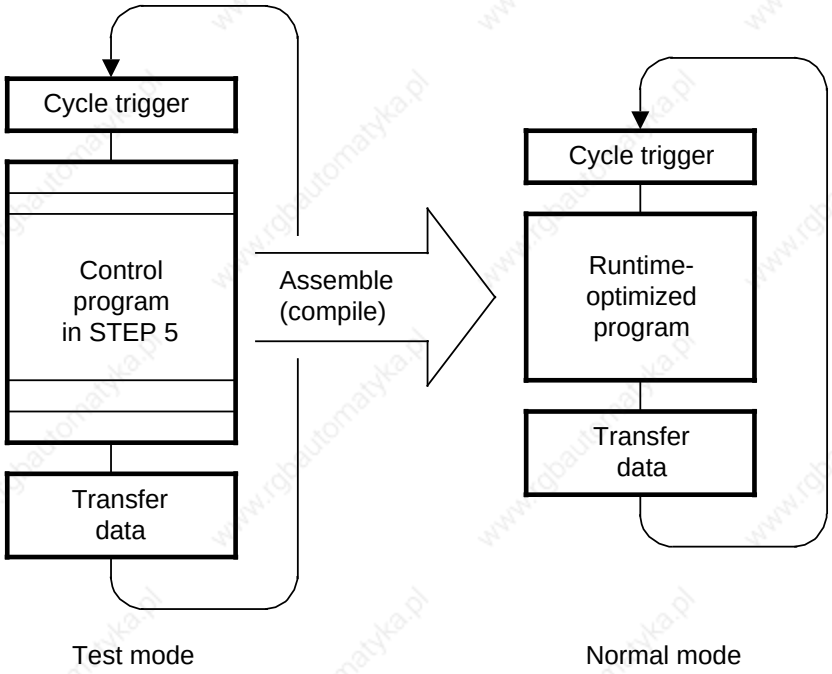



**Figure 7-9. Program Scanning with CPU 102**

## Special Features of the Normal Mode

### Significance of the Memory Submodule

Normal mode is only possible if the memory submodule is plugged in.

This submodule contains only the STEP 5 program.

The CPU RAM contains the STEP 5 program and the compiled program to be processed.

### Program Change

You can enter, modify, or erase PBs, OBs and FBs only in the test mode.

You can read out the STEP 5 program with the programmer.

### Signal Status Display

You can monitor and control signals states with the “STATUS VAR” and “FORCE VAR” functions. The “STATUS” function can be used only in the test mode.

### Diagnostics

The “BSTACK” diagnostics function cannot be activated.

### Fault Analysis

The ISTACK bytes 23 to 27 are not valid. Therefore, you cannot determine the point in a program where an interruption took place (programmable controller in STOP, e.g., programmed loop with timeout). However, when compiling the program, errors (e.g., illegal operations and parameters) are detected and displayed by the STEP address counter in the ISTACK. This counter points to the error in the STEP 5 program.

## Mode Change

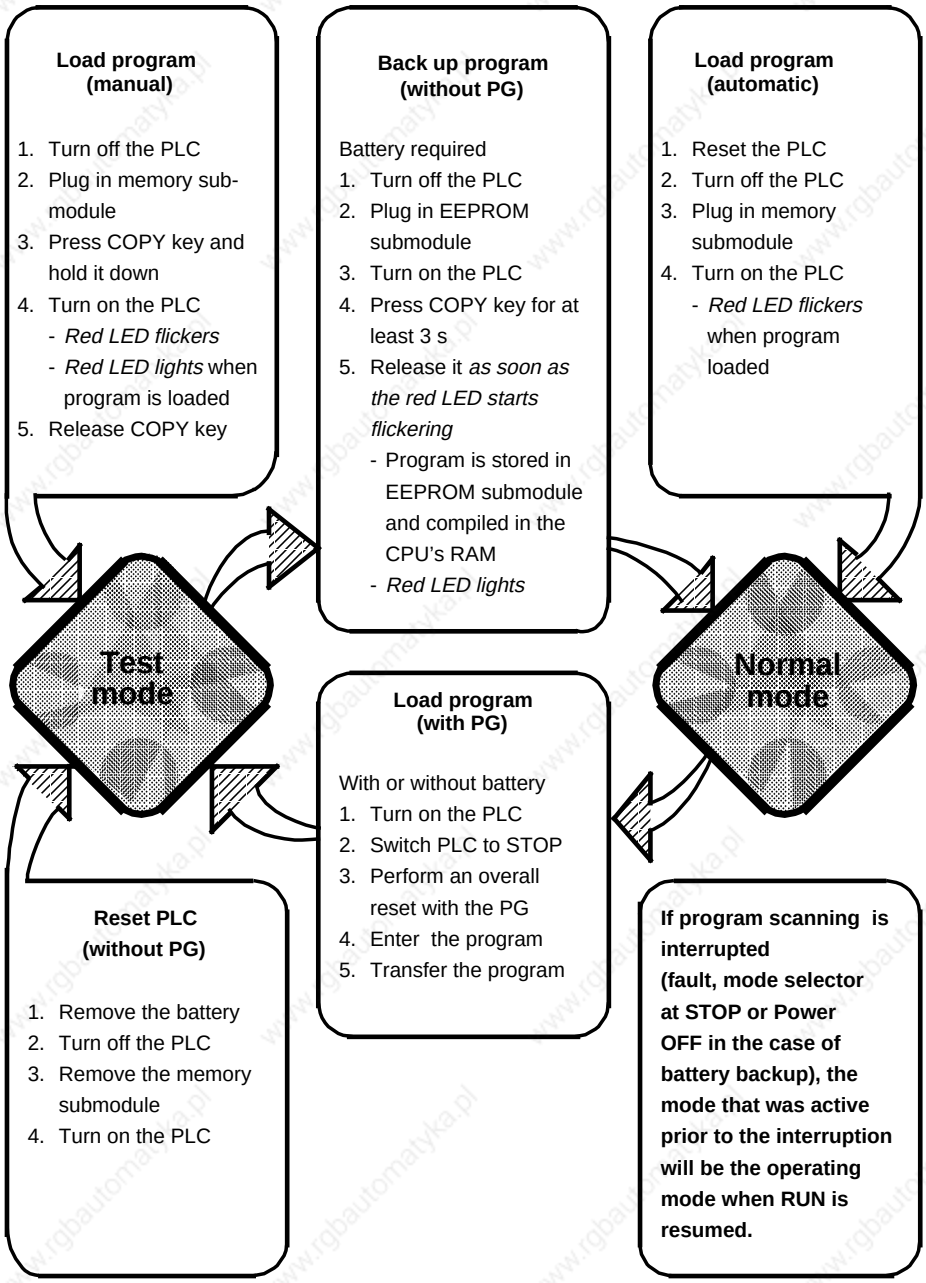



**Figure 7-10. Mode Change for CPU 102**

**Determining the Processing Mode in the ISTACK**

| Byte \ Bit | 7 | 6 | ... |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| : | | | |
| **6** | **KEIN AS** | | |
| 7 | | | |
| : | | | |

**Figure 7-11. Display of the Processing Mode in the ISTACK**

You can use a programmer to check the current processing mode in the ISTACK. The ISTACK display, byte 6, is possible in RUN and STOP (see section 5.2).

**KEIN AS=1 : Test mode**

Execution time is 70 ms/1024 binary statements. There are no limitations on the test or operator functions.

**KEIN AS=0: Normal mode**

Execution time is 7 ms/1024 binary statements. There are limited test and operator functions.

**Further Reduction in the Execution Time in Normal Mode**

Logic operations executed in one input byte, output byte, or flag byte require only 2 µs per logic operation. Program your control according to example 2.

Example 1:

| STL | | | Time/ µs |
|---|---|---|---|
| A | I | 0.0 | 5 |
| AN | I | 1.1 | 6 |
| ON | I | 2.3 | 6 |
| O | I | 3.5 | 6 |
| = | Q | 4.2 | 8 |
| | | | |
| A | F | 15.1 | 5 |
| A | F | 16.3 | 6 |
| AN | F | 17.7 | 6 |
| = | Q | 4.5 | 8 |
| Execution time | | | 56 µs |

Approx. 6 µs/binary operation

Example 2:

| STL | | | Time/ µs |
|---|---|---|---|
| A | I | 0.0 | 5 |
| AN | I | 0.1 | 2 |
| ON | I | 0.3 | 2 |
| O | I | 0.5 | 2 |
| = | Q | 4.2 | 8 |
| | | | |
| A | F | 15.1 | 5 |
| A | F | 15.3 | 2 |
| AN | F | 15.7 | 2 |
| = | Q | 4.5 | 8 |
| Execution time | | | 36 µs |

Approx. 4 µs/binary operation

### 7.4.2 START-UP Program Processing

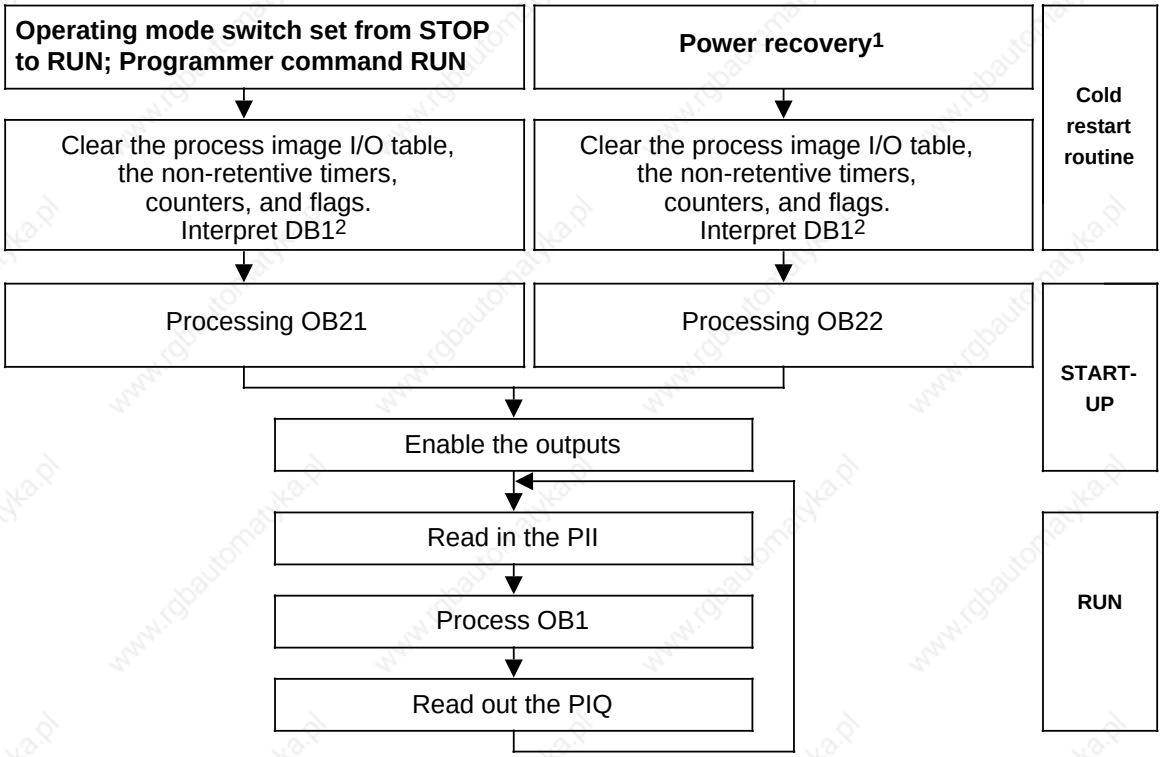
In the START-UP mode, the operating system of the CPU automatically calls up a start-up OB if the OB has been programmed.

- OB21 is called up for a manual cold restart.
- OB22 is called up for an automatic cold start after power recovery if the programmable controller was previously in the RUN mode.

If you have programmed start-up OBs, they are processed before the cyclic program processing occurs. The start-up OB program is appropriate, for example, for a one-time presetting of certain system data. If the appropriate start-up OB is not programmed, the programmable controller jumps directly to the RUN mode. See section 4.1.2.

Features of the start-up blocks (OB21, OB22):

- The red and green LEDs light.
- Timers are processed.
- Scan monitoring is not activated.
- Interrupts are not processed.




1 This is the procedure if the programmable controller was in the RUN mode when the power went off, if the mode switch was still on RUN when the power was restored, and if the battery was inserted. If the battery was not inserted, you must insert a memory submodule containing the valid blocks.

2 For CPU 103 version 8MA03 and higher

**Figure 7-12. Setting the Start-Up Procedure**

The following two examples show you how you can program a start-up OB.

**Example 1:** Programming OB22

| Example | STL | Explanation |
|---|---|---|
| After power recovery, you want to be sure that the power supply voltage for the I/Os has attained its rated value before the cyclic program is processed. A time loop is therefore programmed in OB22. | AN T 1<br>L KT 500.0<br>SP T 1<br>F001: A T 1<br>JC =F001<br>BE | A 5 s time value is loaded in ACCU 1.<br>Timer 1 is started.<br>After 5 s, cyclic program processing begins in OB1. |

**Example 2:** Programming OB21

| Example | STL | Explanation |
|---|---|---|
| After the operating mode switch causes a cold restart, flag bytes 0 to 9 are preset with “0”. The other flag bytes are retained since they contain important machine functions. | L KH 0<br>T FW 0<br>T FW 2<br>T FW 4<br>T FW 6<br>T FW 8<br>BE | Value “0” is loaded in ACCU 1 and transferred into flag words 0, 2, 4, 6, and 8. |

### 7.4.3 Cyclic Program Processing

The operating system calls OB1 cyclically. If you want to have structured programming, you should program only jump operations (block calls) in OB1. The blocks you call up, PBs, FBs, and SBs, should contain completed functional units in order to provide a clearer overview.

A time monitor is triggered at the beginning of each program processing cycle (scan cycle trigger). If the scan cycle time trigger is not reset within the scan monitoring time, the CPU automatically enters the STOP mode and disables the output modules.

You can set the monitoring time (see Table 6-6). You could have a control program that is so complex that it cannot be processed within 300 ms. With CPU 103 and higher, you can use OB31 (see section 9.3) to lengthen (retrigger) the scan monitoring time in the control program.

Monitoring time is exceeded, for example, if you program endless loops or if there is a malfunction in the programmable controller.

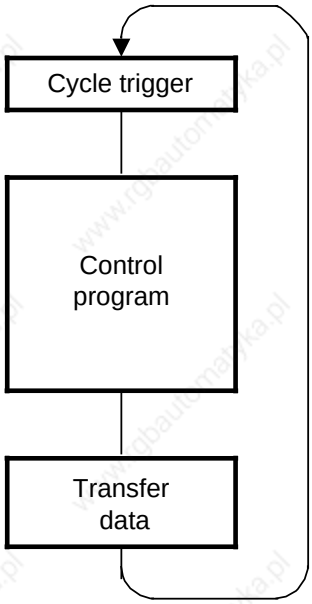



**Figure 7-13. Cyclic Program Processing**

**Response Time**

Response time $t_R$ is defined as the time between a change in the input signal and the subsequent change in the output signal.

Prerequisites for the following information:

- No interrupts are running.
- The programmer interface is not in use. (The load is very dependent on the function.)

The response time is influenced by the following factors:

- The input module delay (see chapter 14)
- The program processing time (see Appendix A)
- The data cycle times (number of data bits x 25 µs - a bus configuration of 256 data bits results in a data cycle time of approximately 8 ms)
- The operating system run time (up to 3% of the program cycle)
- The processing of the internal timers T 0 to T15 for CPU 100<br>T 0 to T31 for CPU 102<br>T 0 to T127 for CPU 103

Calculating the maximum response time $t_{Rm}$:

- With $t_G$= 2 x program processing time + 3 x data cycle time + 3 x operating system run time + delay time of the input modules
- Maximum processing time of the internal timers $t_{Tm}$

  $t_{Tm}$ = number of processed timers x 32 µs (number of processed timers for CPU 100: 16<br>number of processed timers for CPU 102: 32<br>number of processed timers for CPU 103: 128)

  $t_{Tm}$ = 103 µs for CPU 103 version 8MA03

$$t_{Rm} = t_G \left(1 + \frac{t_{Tm}}{10\ ms}\right) + t_{Tm}.$$

During the transition from STOP to RUN, there is a one-time increase in the response time to about 200 ms.

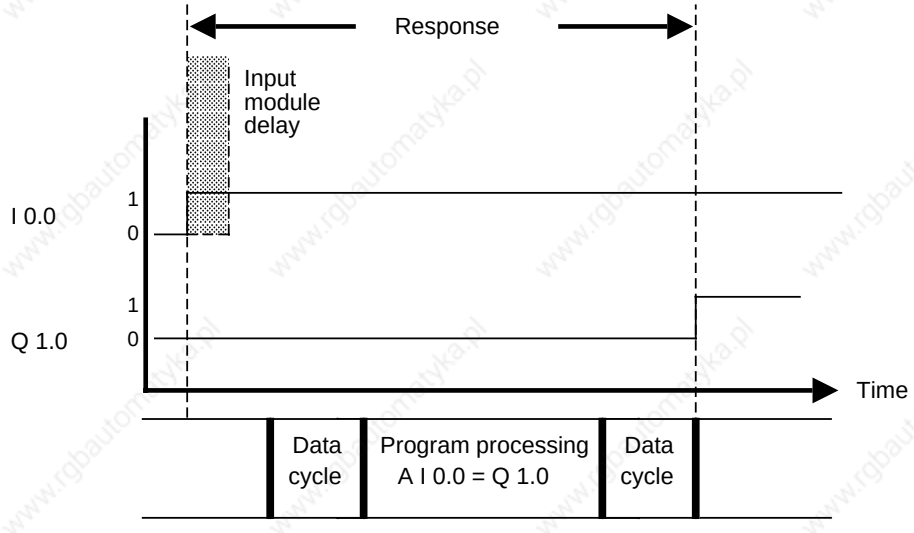



**Figure 7-14. Calculating the Response Time**

### 7.4.4 Time-Controlled Program Processing, for CPU 103 Version 8MA02 and Higher

Time-controlled program processing can be defined as a (periodic) time signal causing the CPU to interrupt cyclic program processing to process a specific program. Once this program has been processed, the CPU returns to the interruption point in the cyclic program and resumes processing.

**Prerequisites for time-controlled program processing**

Time-controlled program processing is possible only if the following prerequisites have been fulfilled.

- Organization block OB13 must be programmed.
- The programmable controller must be set to Power ON and the RUN mode must be selected.
- Interrupt processing may not be disabled (by the IA - disable interrupt - operation). See section 8.2.8.
- The OB13 call-up interval is set to > 0.

OB13 is available for time-controlled program processing when using CPU 103 version 8MA02 and higher. You determine the intervals at which you want the operating system to process OB13. It is also possible to change the call-up intervals during cyclic program processing. Cyclical program processing continues if OB13 is not programmed.

- **Setting the call-up interval**
  You can set the call-up interval in DB1 using the TFB: block ID. You can set the times from 10 ms to 655,530 ms (use 10-ms increments). The default for OB13 is 100 ms.
- **Interrupt possibilities**
  OB13 can interrupt the cyclical program after any STEP 5 statement. After the current STEP 5 statement is executed, you can use process interrupts to interrupt time-controlled program processing. After interrupt processing, time-controlled program processing continues until it is finished.

  OB13 cannot interrupt the operating system, the process interrupts (OB2), or the current time-controlled program processing (OB13).
- **Disabling/enabling the call-up**
  Use the IA command to disable, and the RA command to enable the OB13 call-up.
  A call-up request can be stored during a call-up disable. The default is RA. See section 8.2.8.

- **Saving data**
  If a time-controlled OB uses scratchpad flags that are also used in the cyclic control program, then these scratchpad flags must be saved in a data block during the processing of the time-controlled OB.

> **Note**
>
> When processing OB13, you may not exceed the block nesting depth of 16 levels. When processing with CPU 103 (6ES5 103-8MA03), you may not exceed the block nesting depth of 32 levels.

- **Reading out the interrupt PII**
  When OB13 is called, the signals of the input modules are read into the interrupt PII. The interrupt PII can be scanned in OB13 by means of the L PY 0 to 127 or L PW 0 to 126 load operations (load byte x or word x of the interrupt PII in ACCU 1). There is an interrupt input data cycle prior to time-controlled program processing. The interrupt data cycle time lengthens the response time of the cyclical program processing.

  If other operands are entered, the CPU goes in the STOP mode (see section 5.2.1). This error is indicated in ISTACK by the “NNN” error message.

- **Writing to the interrupt PIQ**
  Data to the external I/Os can be written to the interrupt PIQ by means of transfer operations T PY 0 to 127 or T PW 0 to 126. The “normal” PIQ is written to simultaneously. After OB13 has finished, the data that has been transferred to the interrupt PIQ is output to the peripheral I/Os in an interrupt output data cycle (before “normal” program processing). The interrupt data cycle time lengthens the response time of the cyclical program processing.

> **Note**
>
> The interrupt output data cycle is executed only if the interrupt PIQ has been written to.

### 7.4.5 Interrupt-Driven Program Processing, for CPU 103 Version 8MA02 and Higher

For CPU 103 version 8MA02 and higher, interrupt-driven program processing is initiated when a signal from the process causes the CPU to interrupt the cyclic or time-controlled program processing and execute a specific program. When this program has been scanned, the CPU returns to the point of interruption in the cyclic or time-controlled program and resumes scanning at that point. Chapter 10 contains detailed information about interrupt processing.

## 7.5 Processing Blocks

Earlier sections in this chapter described how to use blocks. Chapter 8 introduces all of the operations required to work with blocks. You can change any block that has been programmed. The following sections will deal only briefly with the different ways you can change blocks. Refer to the operator‘s guide for your programmer for more detailed information on changing blocks.

### 7.5.1 Changing Programs

You can use the following programmer functions to make program changes with any block type.

- INPUT
- OUTPUT
- STATUS (see section 4.5)

These three programmer functions make it possible for you to make the following types of changes:

- Delete, insert, or overwrite statements.
- Insert or delete segments.

### 7.5.2 Changing Blocks

Program changes refer to changing the contents of a block. You can also delete or overwrite a complete block. When you delete a block, it is not deleted from the program memory but simply becomes invalid. You cannot enter new information in the memory location of an invalid block. This may cause new blocks not to be accepted. If a new block is not accepted, then the PG transmits the “no space available” error message. You can make more space by compressing the programmable controller memory.

### 7.5.3 Compressing the Program Memory

Figure 7-15 illustrates what takes place in the program memory during a COMPRESS operation. Internally, one block is shifted per cycle.

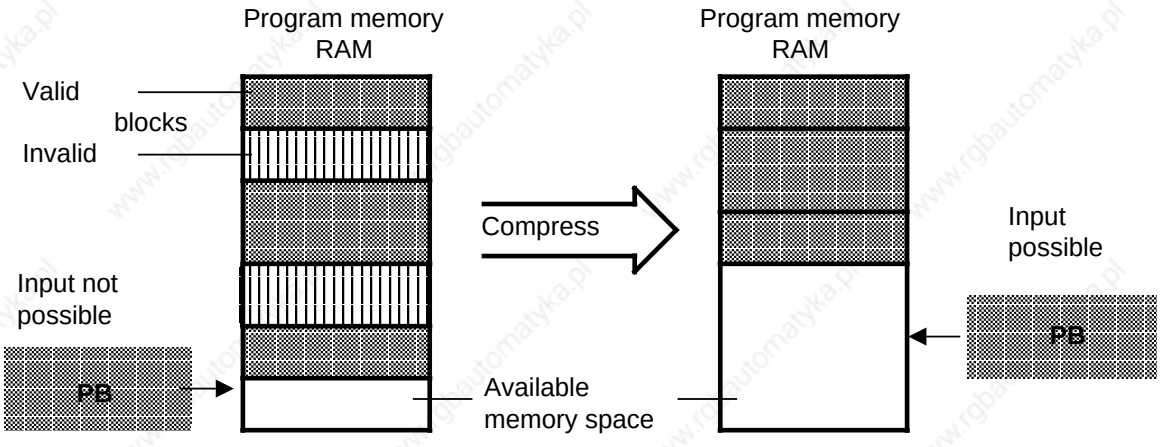



**Figure 7-15. Compressing the Program Memory**

You can use the COMPRESS programmer function to clean up internal program memory.

If there is a power failure during the compress operation when a block is being shifted and block shifting can not be completed, the CPU remains in the STOP mode. The “NINEU” error message appears. Both the“BSTSCH” and the “SCHTAE” bits are set in the ISTACK.
Remedy: Overall reset.

## 7.6 Number Representation

With STEP 5 you can work with numbers in the following five representations:

- Decimal numbers from -32768 to +32767 (KF)
- Hexadecimal numbers from 0000 to FFFF (KH)
- BCD-coded numbers (4 tetrads) from 0000 to 9999
- Bit patterns (KM)
- Constant byte (two-byte representation) from 0 to 255 for each byte (KY)

**Number Formats**

The programmable controller is designed to process binary signal states (only “0” and “1”). Therefore the programmable controller represents all numbers internally as 16-bit binary numbers or as bit patterns.

Four bits can be combined into a tetrad (BCD) to shorten the binary code representation. The value of these tetrads can be displayed in hexadecimal representation.

**Example**: 16-bit binary coded number and shortened hexadecimal representation

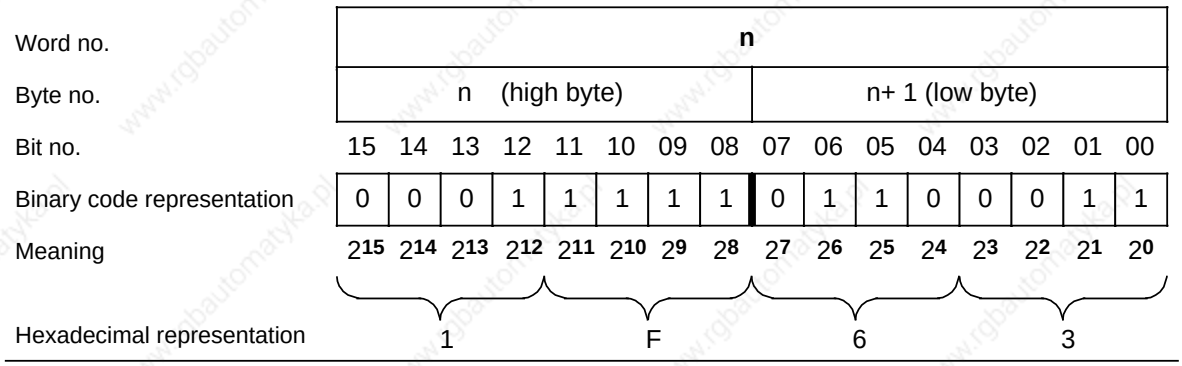



**Figure 7-16. Bit Assignment of a 16-Bit Fixed-Point Binary Number**

You can work with binary-coded decimals to program timers and counters in the decimal system.

BCD tetrads are defined in the range of 0 to 9.

**Example**: 12-bit timer or counter value in BCD and decimal formats

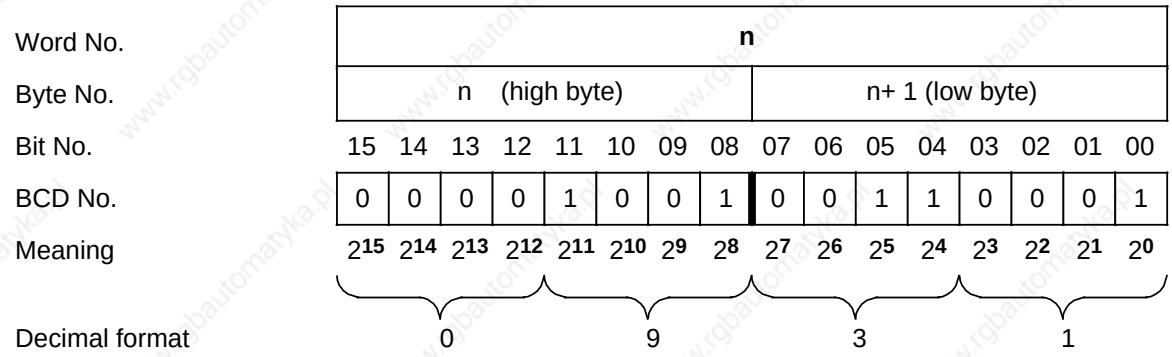



**Figure 7-17. BCD and Decimal Formats**

**Table 7-6. Comparison of Number Formats**

| Binary | Decimal | BCD | Hexadecimal |
|---|---|---|---|
| 0000 | 0 | 0000 0000 | 0 |
| 0001 | 1 | 0000 0001 | 1 |
| 0010 | 2 | 0000 0010 | 2 |
| 0011 | 3 | 0000 0011 | 3 |
| 0100 | 4 | 0000 0100 | 4 |
| 0101 | 5 | 0000 0101 | 5 |
| 0110 | 6 | 0000 0110 | 6 |
| 0111 | 7 | 0000 0111 | 7 |
| 1000 | 8 | 0000 1000 | 8 |
| 1001 | 9 | 0000 1001 | 9 |
| 1010 | 10 | 0001 0000 | A |
| 1011 | 11 | 0001 0001 | B |
| 1100 | 12 | 0001 0010 | C |
| 1101 | 13 | 0001 0011 | D |
| 1110 | 14 | 0001 0100 | E |
| 1111 | 15 | 0001 0101 | F |

You can use the “LC” operation to convert a binary number to a BCD number for timers and counters.

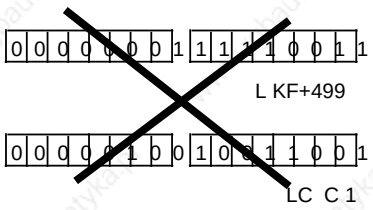
**Example:** Comparing a count in counter 1 with decimal number 499

The comparison value must be stored in the accunulator by means of a load operation. Use the “LKF + 499” statement so that you do not have to convert the value 499 into other numerical systems (binary or hexadecimal) for the input. The number $1F3_H$ is then stored in the accumulator.

The current count must also be loaded into the accumulator.

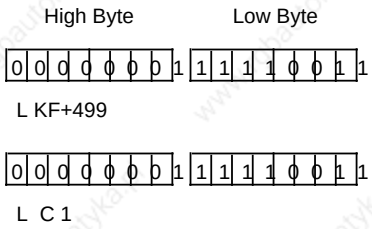
Incorrect Method:

If you use the “LCC1” statement, the current count will be loaded in BCD. The “!=F” comparison operation results in a “not equal to” condition since the comparison uses different formats.




Correct Method:

The formats are identical if the “LC1” statement is input.



## 8 STEP 5 Operations


| | | |
|---|---|---|
| 8.1 | Basic Operations | 8 - 1 |
| 8.1.1 | Boolean Logic Operations | 8 - 2 |
| 8.1.2 | Set/Reset Operations | 8 - 7 |
| 8.1.3 | Load and Transfer Operations | 8 - 10 |
| 8.1.4 | Timer Operations | 8 - 15 |
| 8.1.5 | Counter Operations | 8 - 25 |
| 8.1.6 | Comparison Operations | 8 - 30 |
| 8.1.7 | Arithmetic Operations | 8 - 31 |
| 8.1.8 | Block Call Operations | 8 - 33 |
| 8.1.9 | Other Operations | 8 - 38 |
| 8.2 | Supplementary Operations | 8 - 39 |
| 8.2.1 | Load Operation, for CPU 103 and Higher | 8 - 40 |
| 8.2.2 | Enable Operation, for CPU 103 and Higher | 8 - 41 |
| 8.2.3 | Bit Test Operations, for CPU 103 and Higher | 8 - 42 |
| 8.2.4 | Digital Logic Operations | 8 - 44 |
| 8.2.5 | Shift Operations | 8 - 48 |
| 8.2.6 | Conversion Operations | 8 - 50 |
| 8.2.7 | Decrement/Increment, for CPU 103 and Higher | 8 - 52 |
| 8.2.8 | Disable/Enable Interrupt, for CPU 103 Version 8MA02 and Higher | 8 - 53 |
| 8.2.9 | “DO” Operation, for CPU 103 and Higher | 8 - 54 |
| 8.2.10 | Jump Operations | 8 - 56 |
| 8.2.11 | Substitution Operations, for CPU 103 and Higher | 8 - 58 |
| 8.3 | System Operations, for CPU 103 and Higher | 8 - 64 |
| 8.3.1 | Set Operations | 8 - 64 |
| 8.3.2 | Load and Transfer Operations | 8 - 64 |
| 8.3.3 | Arithmetic Operations | 8 - 67 |
| 8.3.4 | Other Operations | 8 - 68 |
| 8.4 | Condition Code Generation | 8 - 69 |
| 8.5 | Sample Programs | 8 - 71 |
| 8.5.1 | Momentary-Contact Relay/Edge Evaluation | 8 - 71 |
| 8.5.2 | Binary Scaler/Binary Divider | 8 - 71 |
| 8.5.3 | Clock/Clock-Pulse Generator | 8 - 73 |

## Figures


8-1 Accumulator Structure . . . . . 8 - 10
8-2 Execution of the Load Operation . . . . . 8 - 12
8-3 Transferring a Byte . . . . . 8 - 12
8-4 Output of the Current Time (Example) . . . . . 8 - 18
8-5 Outputting the Current Counter Status (Example) . . . . . 8 - 27
8-6 Executing a “DO” Operation . . . . . 8 - 55


## Tables

### Basic Operations


8-1 Overview of Boolean Logic Operations . . . . . 8 - 2
8-2 Overview of the Set/Reset Operations . . . . . 8 - 7
8-3 Overview of Load and Transfer Operations . . . . . 8 - 11
8-4 Overview of Timer Operations . . . . . 8 - 15
8-5 Overview of Counter Operations . . . . . 8 - 25
8-6 Overview of Comparison Operations . . . . . 8 - 30
8-7 Overview of Arithmetic Operations . . . . . 8 - 31
8-8 Overview of Block Call Operations . . . . . 8 - 33
8-9 Other Operations . . . . . 8 - 38


### Supplementary Operations


8-10 Load Operation . . . . . 8 - 40
8-11 Enable Operation . . . . . 8 - 41
8-12 Overview of Bit Operations . . . . . 8 - 42
8-13 Effect of “TB” and “TBN” on the RLO . . . . . 8 - 42
8-14 Overview of Digital Logic Operations . . . . . 8 - 44
8-15 Overview of Shift Operations . . . . . 8 - 48
8-16 Overview of Conversion Operations . . . . . 8 - 50
8-17 Decrement/Increment Operations . . . . . 8 - 52
8-18 Disable/Enable Interrupt Operations . . . . . 8 - 53
8-19 Overview of the “DO” Operation . . . . . 8 - 54
8-20 Overview of Jump Operations . . . . . 8 - 56
8-21 Overview of Binary Logic Operations . . . . . 8 - 58
8-22 Overview of Set/Reset Operations . . . . . 8 - 59
8-23 Overview of Load and Transfer Operations . . . . . 8 - 60
8-24 Overview of Timer and Counter Operations . . . . . 8 - 61
8-25 “DO” Operation . . . . . 8 - 63
8-26 Overview of Set Operations . . . . . 8 - 64


### System Operations


8-27 Overview of Load and Transfer Operations . . . . . 8 - 65
8-28 Overview of the “ADD” Operation . . . . . 8 - 67
8-29 The “TAK” and “STS” Operations . . . . . 8 - 68
8-30 Condition Code Settings for Comparison Operations . . . . . 8 - 69
8-31 Condition Code Settings for Fixed-Point Arithmetic Operations . . . . . 8 - 69
8-32 Condition Code Settings for Digital Logic Operations . . . . . 8 - 70
8-33 Condition Code Settings for Shift Operations . . . . . 8 - 70
8-34 Condition Code Settings for Conversion Operations . . . . . 8 - 70

# 8 STEP 5 Operations

The STEP 5 programming language has the following three operation types:

- Basic Operations include functions that can be executed in organization, program, sequence, and function blocks. Except for the addition (+F), subtraction (-F), and organizational operations, the basic operations can be input and output in the statement list (STL), control system flowchart (CSF), or ladder diagram (LAD) methods of representation.
- Supplementary Operations include complex functions such as substitution statements, test functions, and shift and conversion operations. They can be input and output in STL form only.
- System Operations access the operating system directly. Only an experienced programmer should use them. System operations can be input and output in STL form only.

## 8.1 Basic Operations

Sections 8.1.1 through 8.1.9 use examples to describe the basic operations.

## 8.1.1 Boolean Logic Operations

Table 8-1 provides an overview of Boolean logic operations. Examples follow the table.

**Table 8-1. Overview of Boolean Logic Operations**

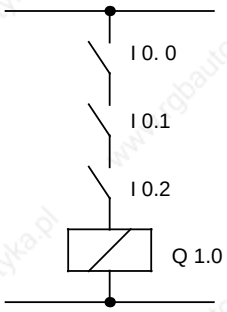
| Operation | Operand | | Meaning |
|---|---|---|---|
| **O** | | | **Combine AND operations through logic OR**<br>Combine the result of the next AND logic operation (RLO) with the previous RLO through logic OR. |
| **A(** | | | **Combine expression enclosed in parentheses through logic AND**<br>Combine the RLO of the expression enclosed in parentheses with the previous RLO through logic AND. |
| **O(** | | | **Combine expression enclosed in parentheses through logic OR**<br>Combine the RLO of the expression enclosed in parentheses with the previous RLO through logic OR. |
| **)** | | | **Close parenthesis**<br>Conclude the expression enclosed in parentheses. |
| **A** | | | **Scan operand for "1" and combine with RLO through logic AND**<br>The result is "1" when the operand in question carries signal state "1". Otherwise the scan results in "0". Combine this result with the RLO in the processor through logic AND[1]. |
| **O** | | | **Scan operand for "1" and combine with RLO through logic OR**<br>The result is "1" when the operand in question has signal state "1". Otherwise the scan results in "0". Combine this result with the RLO in the processor through logic OR[1]. |
| **AN** | | | **Scan operand for "0" and combine with RLO through logic AND**<br>The result is "1" when the operand in question has signal state "0". Otherwise the scan results in "0". Combine this result with the RLO in the processor through logic AND[1]. |
| **ON** | | | **Scan operand for "0" and combine with RLO through logic OR**<br>The result is "1" when the operand in question has signal state "0". Otherwise the scan results in "0". Combine this result with the RLO in the processor through logic OR[1]. |

| ID | Parameter | CPU 100 | CPU 102 | CPU 103 |
|---|---|---|---|---|
| I | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 127.7 |
| Q | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 127.7 |
| F | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 255.7 |
| T | | 0 to 15 | 0 to 31 | 0 to 127 |
| C | | 0 to 15 | 0 to 31 | 0 to 127 |

1 If the scan follows an RLO limiting operation directly (first scan), the scan result is reloaded as a new RLO.

### AND Operation

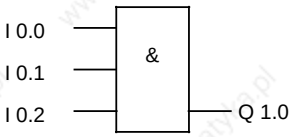
The AND operation scans to see if various conditions are satisfied simultaneously.

| Example | Circuit Diagram |
|---|---|
| Output Q 1.0 is “1” when all three inputs are “1”.<br>The output is “0” if at least one input is “0”.<br>The number of scans and the sequence of the logic statements are at random. |  |

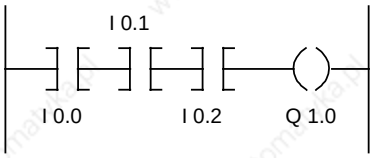
**STL**

```
A     I     0.0
A     I     0.1
A     I     0.2
=     Q     1.0
```

**CSF**

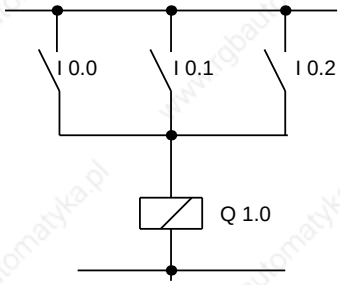



**LAD**




### OR Operation

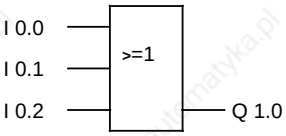
The OR operation scans to see if one of two (or more) conditions has been satisfied.

| Example | Circuit Diagram |
|---|---|
| Output Q 1.0 is “1” when at least one of the inputs is “1”.<br>Output Q 1.0 is “0” when all inputs are “0” simultaneously.<br>The number of scans and the sequence of their programming are optional. |  |

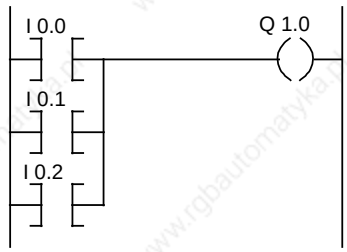
**STL**

```
O     I     0.0
O     I     0.1
O     I     0.2
=     Q     1.0
```

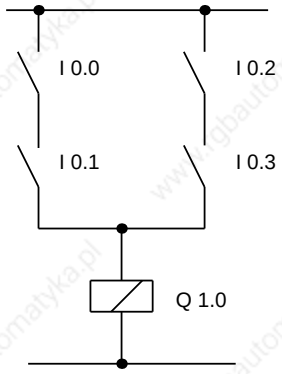
**CSF**

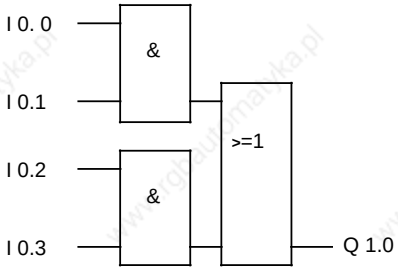
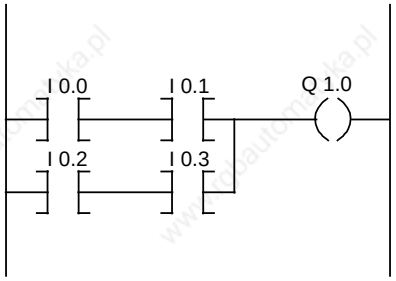



**LAD**

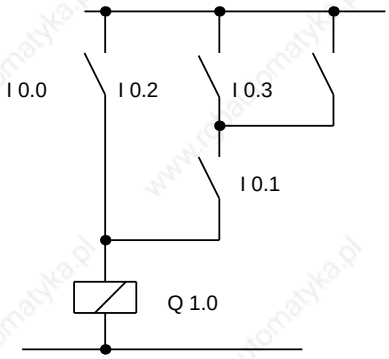


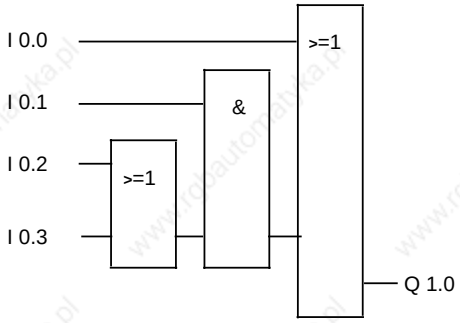
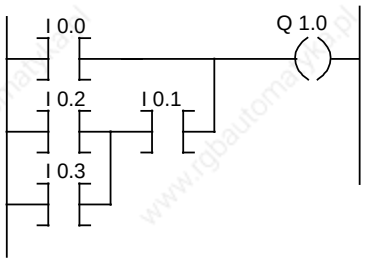
**AND before OR Operation**

| Example | Circuit Diagram |
|---|---|
| Output Q 1.0 is “1” when at least one AND condition has been satisfied.<br>Output Q 1.0 is “0” when neither of the two AND conditions has been satisfied. |  |

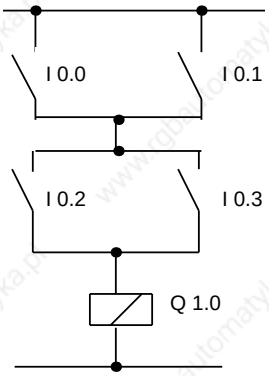
| STL | CSF | LAD |
|---|---|---|
| A I 0.0<br>A I 0 1<br>O<br>A I 0.2<br>A I 0.3<br>= Q 1.0 |  |  |

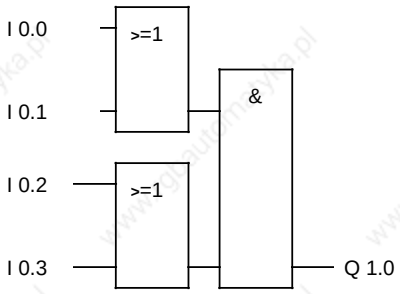
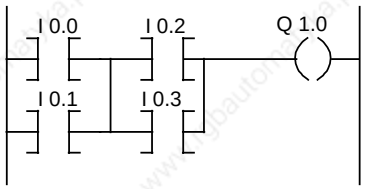
**OR before AND Operation**

| Example | Circuit Diagram |
|---|---|
| Output Q 1.0 is “1” when one of the following conditions has been satisfied:<br>• Input I 0.0 is “1”.<br>• Input I 0.1 and either input I 0.2 or I 0.3 is “1”.<br><br>Output Q 1.0 is “0” when none of the AND conditions has been satisfied. |  |

| STL | CSF | LAD |
|---|---|---|
| O I 0.0<br>O<br>A I 0.1<br>A(<br>O I 0.2<br>O I 0.3<br>)<br>= Q 1.0 |  |  |

**OR before AND Operation**

| Example | Circuit Diagram |
|---|---|
| Output Q 1.0 is “1” when both OR conditions have been satisfied.<br>Output Q 1.0 is “0” when at least one OR condition has not been satisfied. |  |

| STL | CSF | LAD |
|---|---|---|
| A(<br>O I 0.0<br>O I 0.1<br>)<br>A(<br>O I 0.2<br>O I 0.3<br>)<br>= Q 1.0 |  |  |

## 8.1.2 Set/Reset Operations

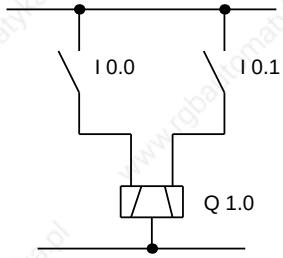
Set/reset operations store the result of logic operation (RLO) formed in the processor. The stored RLO represents the signal state of the addressed operand. Storage can be dynamic (assignment) or static (set and reset). Table 8-2 provides an overview of the set/reset operations. Examples follow the table.

**Table 8-2. Overview of the Set/Reset Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **S** | | | **Set**<br>The first time the program is scanned with RLO = “1”, signal state “1” is assigned to the addressed operand. An RLO change does not affect this status. |
| **R** | | | **Reset**<br>The first time the program is scanned with RLO = “1”, signal state “0” is assigned to the addressed operand. An RLO change does not affect this status. |
| **=** | | | **Assign**<br>Every time the program is scanned, the current RLO is assigned to the addressed operand. |

| ID | Parameter | CPU 100 | CPU 102 | CPU 103 |
|---|---|---|---|---|
| I | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 127.7 |
| Q | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 127.7 |
| F | | 0.0 to 127.7 | 0.0 to 127.7 | 0.0 to 255.7 |

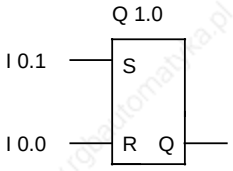
**Flip-Flop for a Latching Signal Output (reset dominant)**

| Example | Circuit Diagram |
|---|---|
| A “1” at input I 0.1 sets flip-flop Q 1.0 (signal state “1”). If the signal state at input I 0.1 changes to “0”, the state of output Q 1.0 is maintained, i.e., the signal is latched.<br>A “1” at input I 0.0 resets the flip-flop (signal state “0”).<br>When the “SET” signal (input I 0.1) and the “RESET” signal (input I 0.0) are applied at the same time, the scanning operation that was programmed last (in this case A I 0.0) is in effect during processing of the rest of the program.<br>In this example, resetting output Q 1.0 has priority. |  |

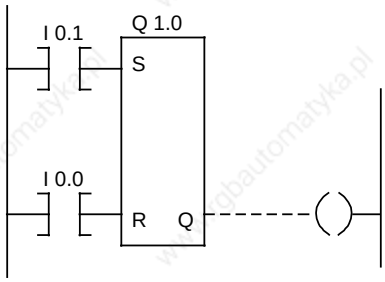
**STL**

```
A     I    0.1
S     Q    1.0
A     I    0.0
R     Q    1.0
NOP   0    *
```

**CSF**

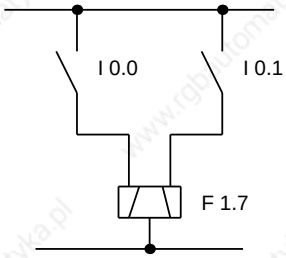
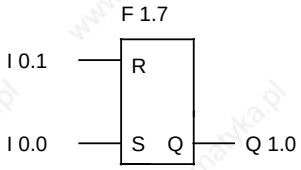
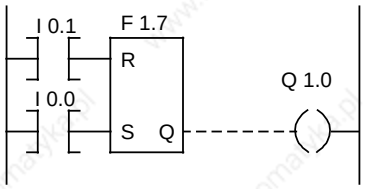



**LAD**




* **NOP 0** “NOP 0” is necessary if the program is to be represented in LAD or CSF form on programmers with a screen. During programming in LAD and CSF, such “NOP 0” operations are allotted automatically.

**RS Flip-Flop with Flags (set dominant)**

<table>
<tr><th colspan="2">Example</th><th>Circuit Diagram</th></tr>
<tr><td colspan="2">A “1” at input I 0.0 sets flip-flop F 1.7 (signal state “1”).<br>If the signal state at input I 0.0 changes to “0”, the state of flag F 1.7 is maintained, i.e., the signal is latched.<br>A “1” at input 0.1 resets the flip-flop (signal state “0”). If the signal state at input I 0.1 changes to “0”, flag F 1.7 retains signal state “0”.<br>If both inputs have a “1” signal state, the flip-flop is set (set dominant).<br>The signal state of the flag is scanned and transferred to output Q 1.0.</td><td> 
</td></tr>
<tr><th>STL</th><th>CSF</th><th>LAD</th></tr>
<tr><td>A I 0.1<br>R F 1.7<br>A I 0.0<br>S F 1.7<br>A F 1.7<br>= Q 1.0</td><td> 
</td><td> 
</td></tr>
</table>

### 8.1.3 Load and Transfer Operations

Use load and transfer operations to do the following tasks.

- Exchange information between various operand areas
- Prepare time and count values for further processing
- Load constants for program processing

Information flows indirectly via accumulators (ACCU 1 and ACCU 2). The accumulators are special registers in the programmable controller that serve as temporary storage. They are each 16 bits long. The accumulators are structured as shown in Figure 8-1.

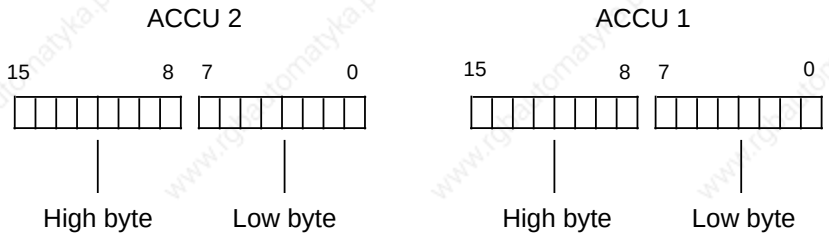



**Figure 8-1. Accumulator Structure**

You can load and transfer permissible operands in bytes or words. For exchange in bytes, information is stored right-justified, i.e., in the low byte.
The remaining bits are set to zero.
You can use various operations to process the information in the two accumulators.

Load and transfer operations are executed independently of condition codes. Execution of these operations does not affect the condition codes.
You can program load and transfer operations graphically only in combination with timer or counter operations; otherwise you can represent them only in STL form.

Table 8-3 provides an overview of the load and transfer operations. Examples follow the table.

**Table 8-3. Overview of Load and Transfer Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **L** | | **Load**<br>The operand contents are copied into ACCU 1 regardless of the RLO.<br>The RLO is not affected. |
| **T** | | **Transfer**<br>The contents of ACCU 1 are assigned to an operand regardless of the RLO.<br>The RLO is not affected. |

| ID | Parameter | | |
|---|---|---|---|
| | **CPU 100** | **CPU 102** | **CPU 103** |
| IB | 0 to 127 | 0 to 127 | 0 to 127 |
| IW | 0 to 126 | 0 to 126 | 0 to 126 |
| QB | 0 to 127 | 0 to 127 | 0 to 127 |
| QW | 0 to 126 | 0 to 126 | 0 to 126 |
| FY | 0 to 127 | 0 to 127 | 0 to 255 |
| FW | 0 to 126 | 0 to 126 | 0 to 254 |
| DR | 0 to 255 | 0 to 255 | 0 to 255 |
| DL | 0 to 255 | 0 to 255 | 0 to 255 |
| DW | 0 to 255 | 0 to 255 | 0 to 255 |
| T[1] | 0 to 15 | 0 to 31 | 0 to 127 |
| C[1] | 0 to 15 | 0 to 31 | 0 to 127 |
| PY | ----- | ----- | 0 to 127 |
| PW | ----- | ----- | 0 to 126 |
| KM[1] | random bit pattern (16 bits) | random bit pattern (16 bits) | random bit pattern (16 bits) |
| KH[1] | 0 to FFFF | 0 to FFFF | 0 to FFFF |
| KF[1] | -32768 to+32767 | -32768 to +32767 | -32768 to +32767 |
| KY[1] | 0 to 255 per byte | 0 to 255 per byte | 0 to 255 per byte |
| KB[1] | 0 to 255 | 0 to 255 | 0 to 255 |
| KS[1] | any 2 alphanumeric characters | any 2 alphanumeric characters | any 2 alphanumeric characters |
| KT[1] | 0.0 to 999.3 | 0.0 to 999.3 | 0.0 to 999.3 |
| KC[1] | 0 to 999 | 0 to 999 | 0 to 999 |

| Operation | Operand | Meaning |
|---|---|---|
| **LD** | | **Load in BCD**<br>Binary times and counts are loaded into ACCU 1 in BCD code regardless of the RLO. |

| ID | Parameter | CPU 100 | CPU 102 | CPU 103 |
|---|---|---|---|---|
| T | | 0 to 15 | 0 to 31 | 0 to 127 |
| C | | 0 to 15 | 0 to 31 | 0 to 127 |

1 These operands cannot be used for transfer.

### Load Operation

During loading, information is copied from a memory area, e.g., from the PII, into ACCU 1.
The previous contents of ACCU 1 are shifted to ACCU 2.
The original contents of ACCU 2 are lost.

**Example:** Two consecutive bytes (IB7 and IB8) are loaded from the PII into the accumulator. Loading does not change the PII (see Figure 8-2).

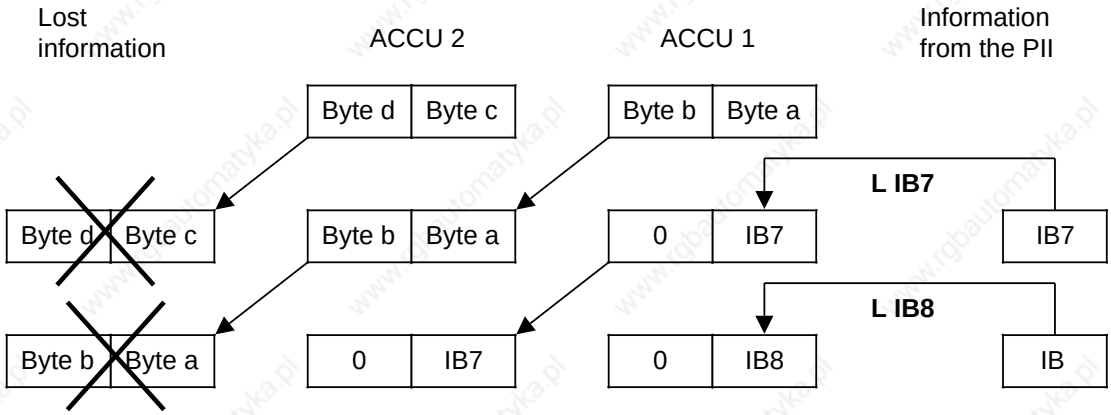



**Figure 8-2. Execution of the Load Operation**

### Transfer Operation

During transfer, information from ACCU 1 is copied into the addressed memory area, e.g., into the PIQ.
This transfer does not affect the contents of ACCU 1.

**Example:** Figure 8-3 shows how byte a, the low byte in ACCU 1, is transferred to QB5.

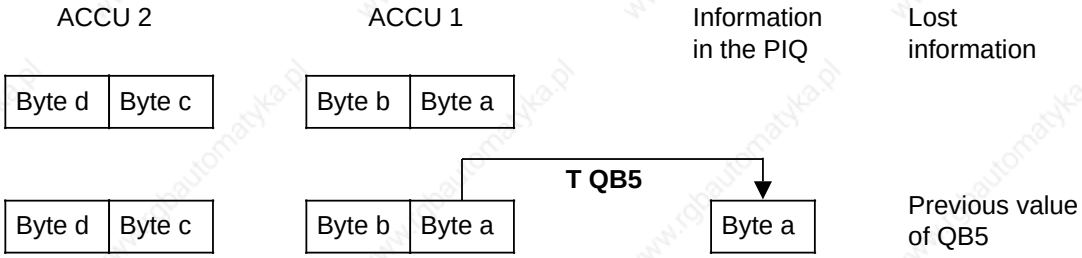
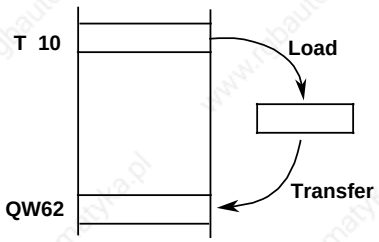



**Figure 8-3. Transferring a Byte**

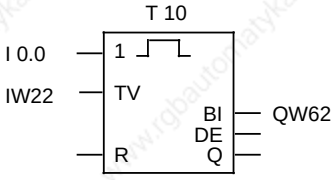
**Loading and Transferring a Time** (See also Timer and Counter Operations)

| Example | Representation |
|---|---|
| During graphic input, QW62 is assigned to output BI of a timer. The programmer automatically stores the corresponding load and transfer operation in the control program. Thus the contents of the memory location addressed with T 10 are loaded into ACCU 1. Afterwards, the contents of the accumulator are transferred to the process image addressed with QW62. In this example, you can see timer T 10 at QW62 in binary code.<br>Outputs BI and DE are digital outputs. The time at output BI is in binary code. The time at output DE is in BCD code with time base. |  |

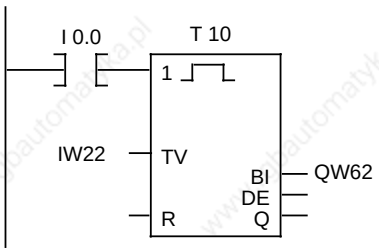
**STL**

```
A     I    0.0
L     IW   22
SP    T    10
NOP 0
L     T    10
T     QW   62
NOP 0
NOP 0
```

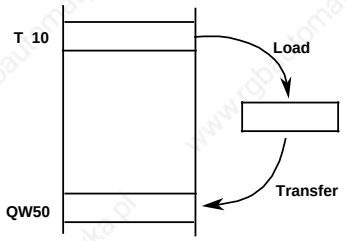
**CSF**

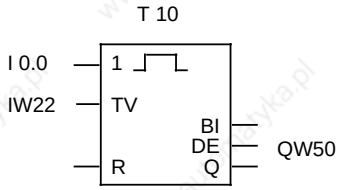
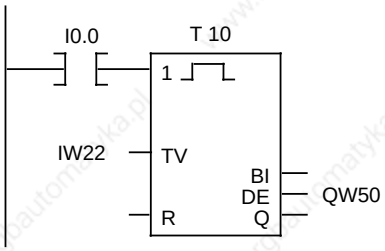



**LAD**



**Loading and Transferring a Time (Coded)**

| Example | Representation |
|---|---|
| The contents of the memory location addressed with T 10 are loaded into the accumulator in BCD code. Then a transfer operation transfers the accumulator contents to the process image memory location addressed by QW50. A coding operation is possible only indirectly for the graphic representation forms LAD and CSF by assigning an address to output DE of a timer or counter location. However, this operation can be entered with a separate statement with STL. |  |

| STL | CSF | LAD |
|---|---|---|
| A I 0.0<br>L IW 22<br>SP T 10<br>NOP 0<br>NOP 0<br>LD T 10<br>T QW 50<br>NOP 0 |  |  |

### 8.1.4 Timer Operations

The program uses timer operations to implement and monitor chronological sequences. Table 8-4 provides an overview of timer operations. Examples follow the table.

**Table 8-4. Overview of Timer Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **SP** | | | **Pulse Timer**<br>The timer is started on the leading edge of the RLO.<br>When the RLO is “0”, the timer is set to “0”.<br>Scans result in signal state “1” as long as the timer is running. |
| **SE** | | | **Extended Pulse Timer**<br>The timer is started on the leading edge of the RLO.<br>When the RLO is “0”, the timer is not affected.<br>Scans result in signal state “1” as long as the timer is running. |
| **SD** | | | **On-Delay Timer**<br>The timer is started on the leading edge of the RLO.<br>When the RLO is “0”, the timer is set to “0”.<br>Scans result in signal state “1” when the timer has run out and the RLO is still pending at the input. |
| **SS** | | | **Stored On-Delay Timer**<br>The timer is started on the leading edge of the RLO.<br>When the RLO is “0”, the timer is not affected.<br>Scans result in signal state “1” when the timer has run out.<br>The signal state becomes “0” when the timer is reset with the “R” operation. |
| **SF** | | | **Off-Delay Timer**<br>The timer is started on the trailing edge of the RLO.<br>When the RLO is “1”, the timer is set to its initial value.<br>Scans result in signal state ”1” as long as the RLO at the input is “1” or the timer is still running. |
| **R** | | | **Reset Timer**<br>The timer is reset to its initial value as long as the RLO is “1”.<br>When the RLO is “0”, the timer is not affected.<br>Scans result in signal state “0” as long as the timer is reset or has not been started yet. |
| | **ID**<br>T | **Parameter** | **CPU 100**<br>0 to 15 **CPU 102**<br>0 to 31 **CPU 103**<br>0 to 127 |

### Loading a Time

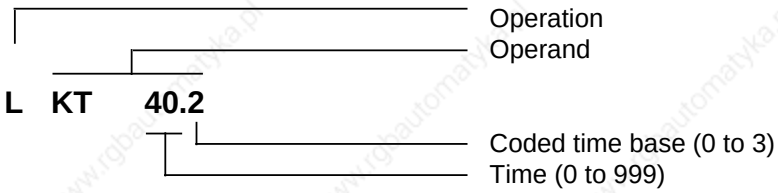
Timer operations call internal timers.
When a timer operation is started, the word in ACCU 1 is used as a time value. You must therefore first specify time values in the accumulator.

You can load a timer with any of the following data types:

| | | |
|---|---|---|
| **KT** | constant time value | |
| | or | |
| **DW** | data word | These data types must be in BCD code. |
| **IW** | input word | |
| **QW** | output word | |
| **FW** | flag word | |

### Loading a Constant Time Value

The following example shows how you can load a time value of 40 s.




### Key for Time Base

| Base | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Factor | 0.01 s | 0.1 s | 1 s | 10 s |

**Example:** KT 40.2 corresponds to 40 x 1 s.

Tolerance:

The time tolerance is equivalent to the time base.

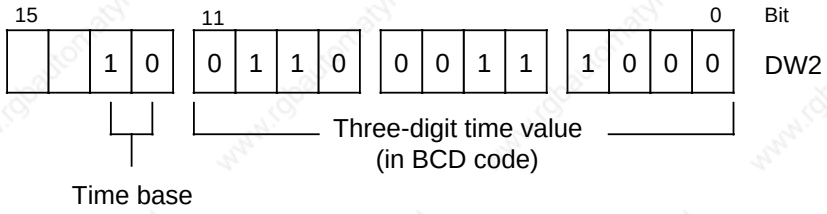
| Examples | Operand | Time Interval |
|---|---|---|
| Possible settings for the time 40 s | KT 400.1 | 400 x 0.1 s - 0.1 s 39.9 s to 40 s |
| | KT 40.2 | 40 x 1 s - 1 s 39 s to 40 s |
| | KT 4.3 | 4 x 10 s - 10 s 30 s to 40 s |

> **Note**
>
> Always use the smallest time base possible.

**Loading a Time as an Input, Output, Flag, or Data Word**

**Load Statement: L DW 2**

The time 638 s is stored in data word DW2 in BCD code.
Bits 14 and 15 are insignificant for the time value.




**Key for Time Base:**

| Base | 0 0 | 0 1 | 1 0 | 1 1 |
|---|---|---|---|---|
| Factor | 0.01 s | 0.1 s | 1 s | 10 s |

You can also use the control program to write to data word DW2.

**Example:** Store the value 270 x 100 ms in data word DW2 of data block DB3.

```
C   DB 3
L   KT 270.1
T   DW2
```

**Output of the Current Time [1]**

You can use a load operation to put the current time into ACCU 1 and process it further from there (see Figure 8-4).
Use the “Load in BCD” operation for digital display output.

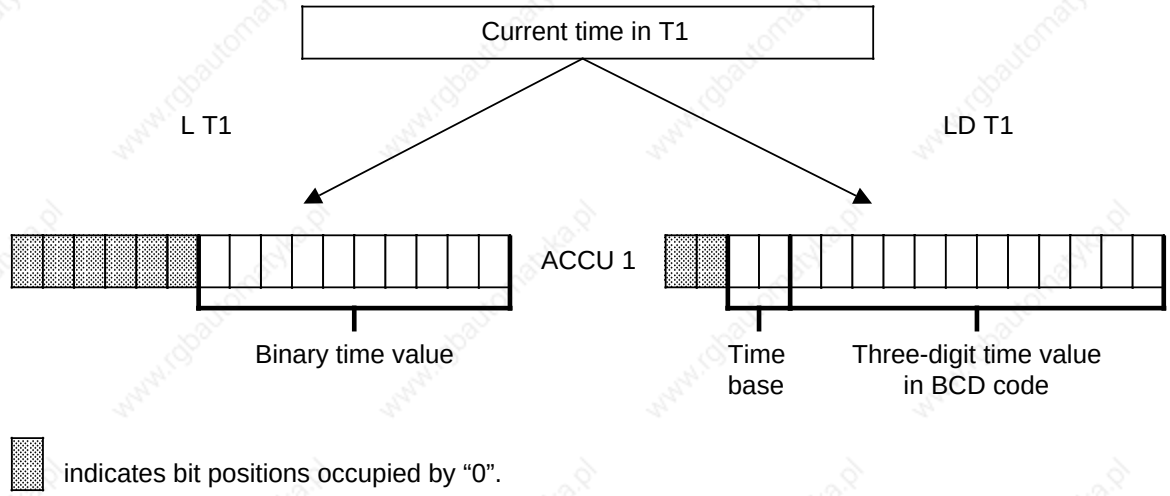


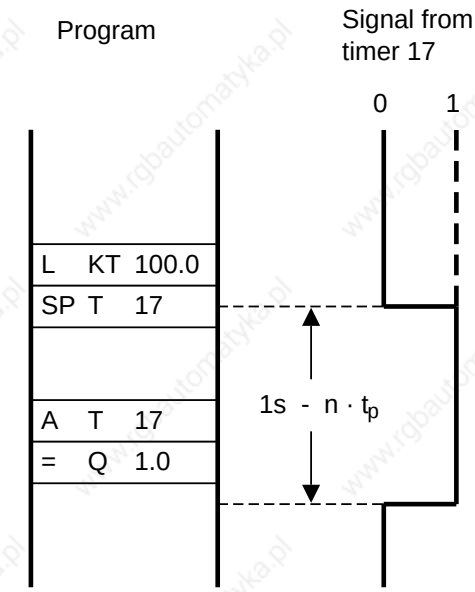

**Figure 8-4. Output of the Current Time (Example)**

[1] The current time is the time value in the addressed timer.

**Starting a timer**

In the programmable controller, timers run asynchronously to program scanning. The time that has been set can run out during a program scanning cycle. It is evaluated by the next time scan. In the worst case, an entire program scanning cycle can go by before this evaluation. Consequently, timers should not activate themselves.

**Example:**

| Schematic Representation | Explanation |
|---|---|
|  <br>n: number of program scanning cycles<br>$t_p$: program scan time | The schematic shows the "$n^{th}$ + 1" processing cycle since timer T 17* was started. Although the timer ran out shortly after the statement "= Q 1.0", output Q 1.0 remains set. The change is not considered until the next program scanning cycle.<br><br>* KT 100.0 is equal to 1 s. |

The following rules apply to timers:

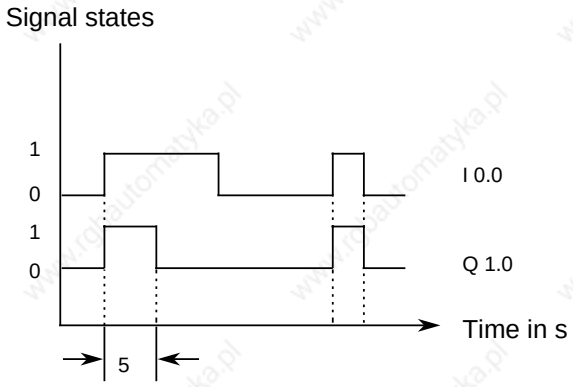
- Except for "Reset timer", all timer operations are started only when there is an edge change. The RLO alternates between "0" and "1".
- After being started, the loaded time is decremented in units corresponding to the time base until it reaches zero.
- If there is an edge change while the timer is running, the timer is reset to its initial value and restarted.
- The signal state of a timer can be scanned with Boolean logic operations.

**Pulse**

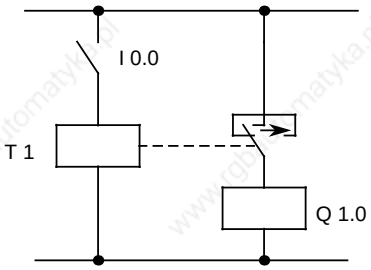
**Example:**

Output Q 1.0 is set when the signal state at input I 0.0 changes from “0” to “1”.
However, the output should not remain set longer than 5 s.

**Timing Diagram**




**Circuit Diagram**

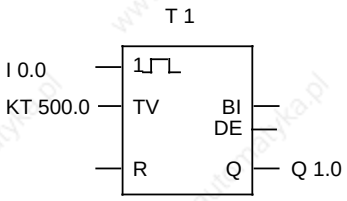



T 1: Time relay with transitional NO contact

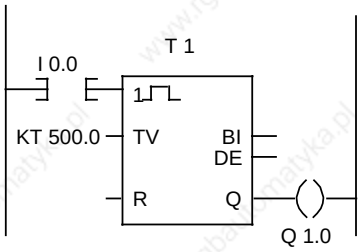
**STL**

```
A     I    0.0
L     KT  500.0
SP    T    1
NOP 0
NOP 0
NOP 0
A     T    1
=     Q    1.0
```

**CSF**




**LAD**




## Note

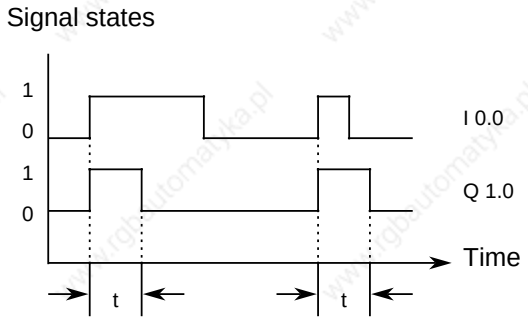
The time tolerance is equivalent to the time base. Always use the smallest time base possible.

**Extended pulse**

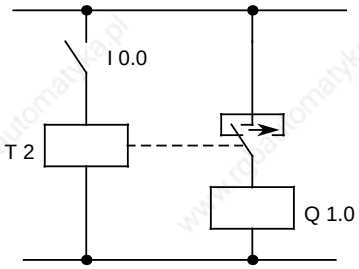
**Example:**

Output Q 1.0 is set for a specific time when the signal at input I 0.0 changes to “1”. The time is indicated in IW16.

**Timing Diagram**




**Circuit Diagram**

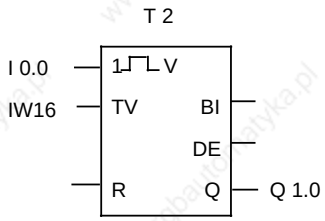



T 2: Time relay with pulse shaper

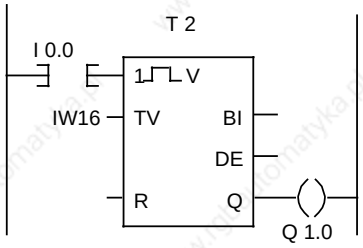
**STL**

```
A     I    0.0
L     IW   16
SE    T    2
NOP 0
NOP 0
NOP 0
A     T    2
=     Q    1.0
```

**CSF**




**LAD**

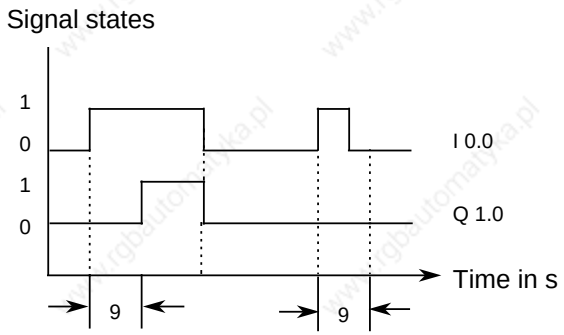


**On-Delay**

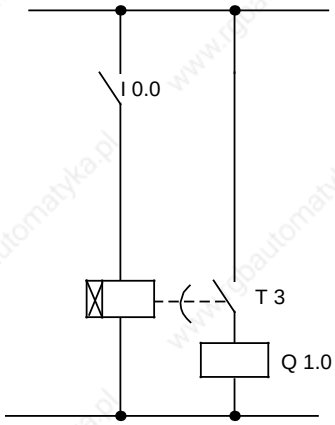
**Example:**

Output Q 1.0 is set 9 s after input I 0.0 and remains set as long as the input carries signal “1”.

**Timing Diagram**



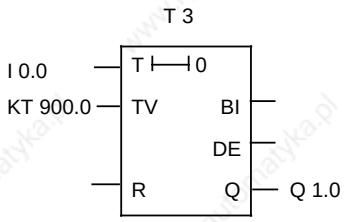

**Circuit Diagram**



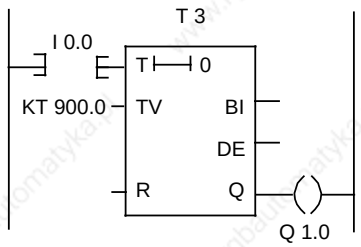

**STL**

```
A     I     0.0
L     KT  900.0
SD    T   3
NOP 0
NOP 0
NOP 0
A     T   3
=     Q     1.0
```

**CSF**




**LAD**

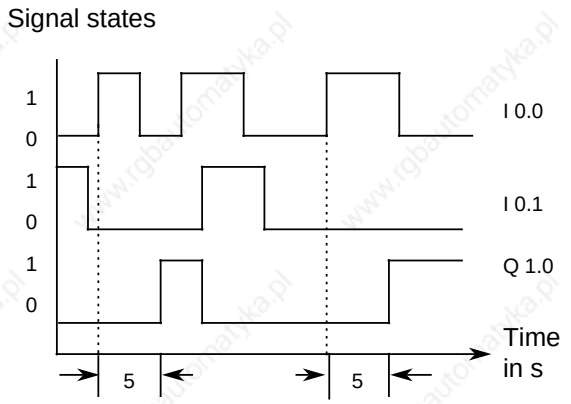


**Stored On-Delay and Reset**

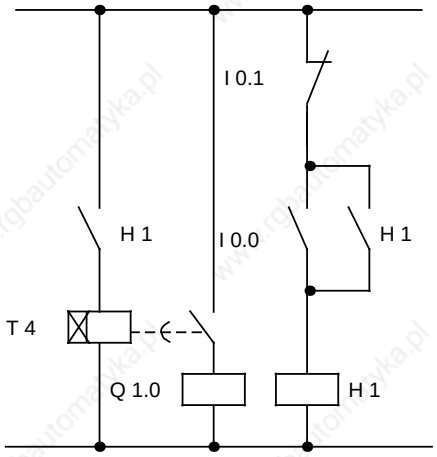
**Example:**

Output Q 1.0 is set 5 s after I 0.0.
Further changes in the signal state at input I 0.0 do not affect the output.
Input I 0.1 resets timer T 4 to its initial value and sets output Q 1.0 to zero.

**Timing Diagram**




**Circuit Diagram**

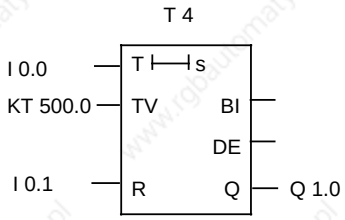



H 1: Auxiliary relay

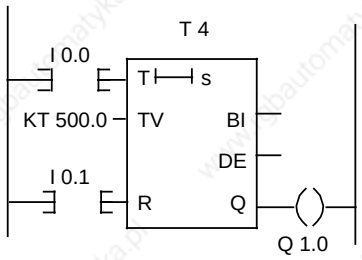
**STL**

```
A     I    0.0
L     KT   500.0
SS    T    4
A     I    0.1
R     T    4
NOP 0
NOP 0
A     T    4
=     Q    1.0
```

**CSF**




**LAD**




**Note**

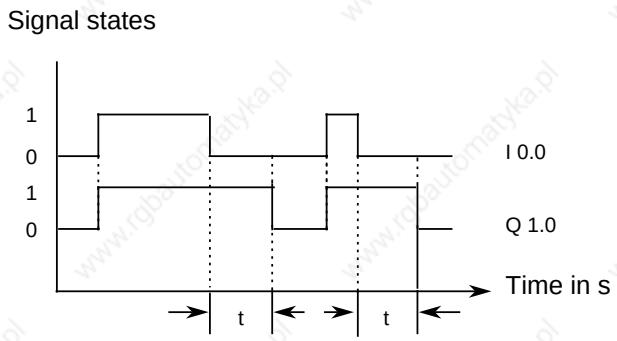
The time tolerance is equivalent to the time base.

**Off-Delay**

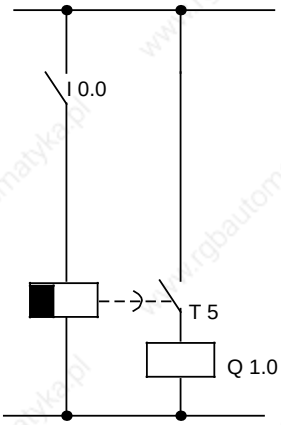
**Example:**

When input I 0.0 is reset, output Q 1.0 is set to zero after a certain delay (t). The value in FW14 specifies the delay time.

**Timing Diagram**



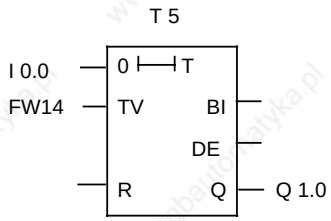

**Circuit Diagram**



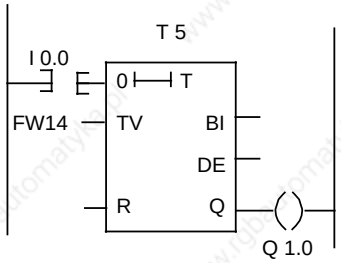

**STL**

```
A     I    0.0
L     FW   14
SF    T    5
NOP 0
NOP 0
NOP 0
A     T    5
=     Q    1.0
```

**CSF**




**LAD**



## 8.1.5 Counter Operations

The programmable controller uses counter operations to handle counting jobs. Counters can count up and down. The counting range is from 0 to 999 (three decades). Table 8-5 provides an overview of the counter operations. Examples follow the table.

**Table 8-5. Overview of Counter Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **S** | | | **Set Counter**<br>The counter is set on the leading edge of the RLO. |
| **R** | | | **Reset Counter**<br>The counter is set to zero as long as the RLO is “1”. |
| **CU** | | | **Count Up**<br>The count is incremented by 1 on the leading edge of the RLO.<br>When the RLO is “0”, the count is not affected. |
| **CD** | | | **Count Down**<br>The count is decremented by 1 on the leading edge of the RLO.<br>When the RLO is “0”, the count is not affected. |
| | **ID**<br>C | **Parameter** | **CPU 100** 0 to 15 **CPU 102** 0 to 31 **CPU 103** 0 to 127 |

### Loading a Count

Counter operations call internal counters.
When a counter is set, the word in ACCU 1 is used as a count. You must therefore first store counts in the accumulator.

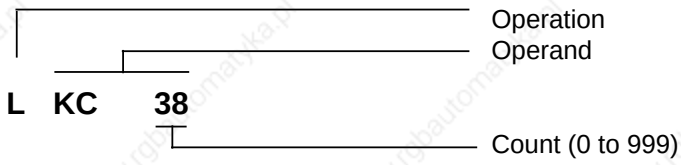
You can load a count with any of the following data types:

**KC** constant count
or
**DW** data word
**IW** input word
**QW** output word
**FW** flag word

} The data for these words must be in BCD code.

### Loading a Constant Count

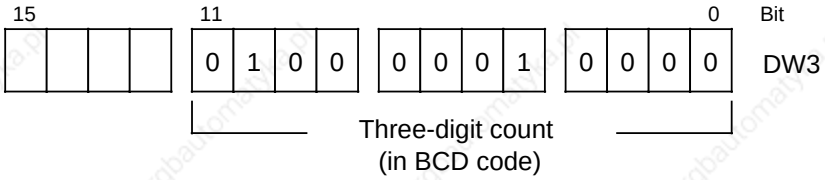
The following example shows how the count 38 is loaded.




### Loading a Count as an Input, Output, Flag, or Data Word

Load statement: **L DW 3**

The count 410 is stored in data word DW3 in BCD code.
Bits 12 to 15 are insignificant for the count.

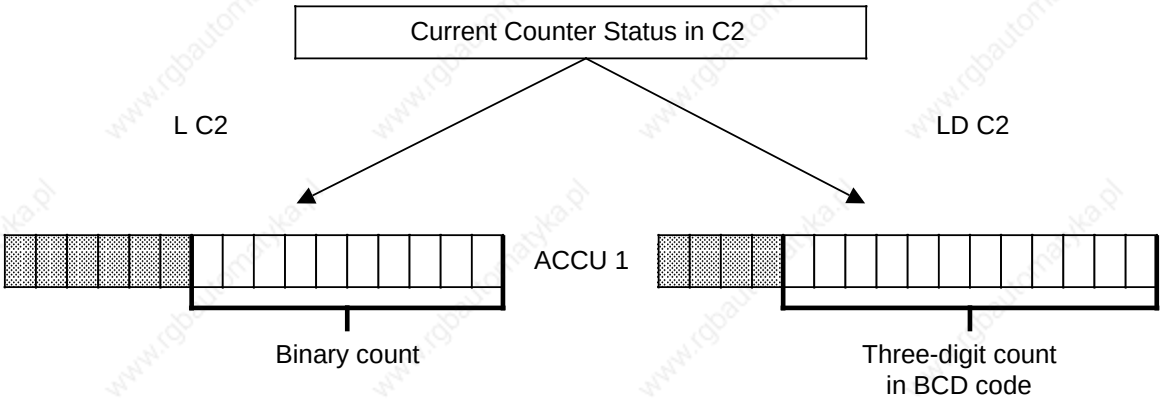



### Scanning the Counter

Use Boolean logic operations to scan the counter status (e.g., A Cx). As long as the count is not zero, the scan result is signal state “1”.

### Outputting the Current Counter Status

You can use a load operation to put the current counter status into ACCU 1 and process it further from there. The “Load in BCD” operation outputs a digital display (see Figure 8-5).




indicates bit positions occupied by “0”.

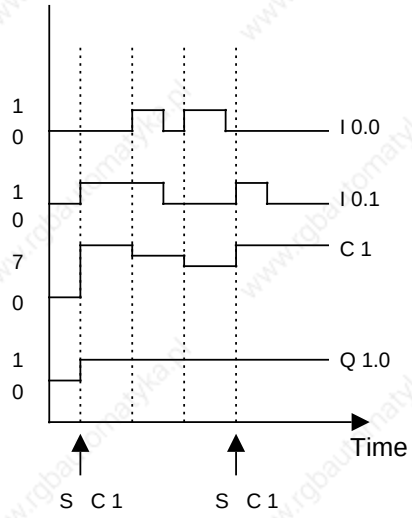
**Figure 8-5. Outputting the Current Counter Status (Example)**

**Setting a Counter “S” and Counting Down “CD”**

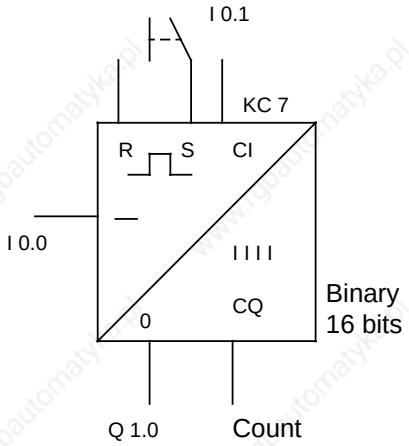
**Example:**

When input I 0.1 is switched on (set), counter 1 is set to count 7. Output Q 1.0 is now “1”.
Every time input I 0.0 is switched on (count down), the count is decremented by 1.
The output is set to “0” when the count is “0”.

**Timing Diagram**



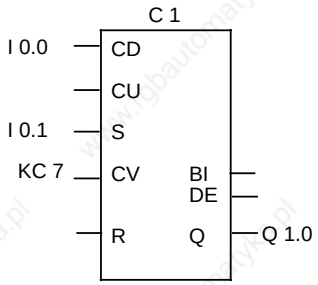

**Circuit Diagram**



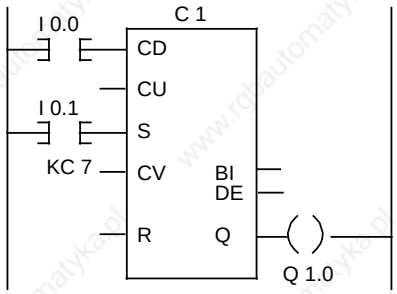

**STL**

```
A     I    0.0
CD    C    1
NOP   0
A     I    0.1
L     KC   7
S     C    1
NOP   0
NOP   0
NOP   0
A     C    1
=     Q    1.0
```

**CSF**




**LAD**

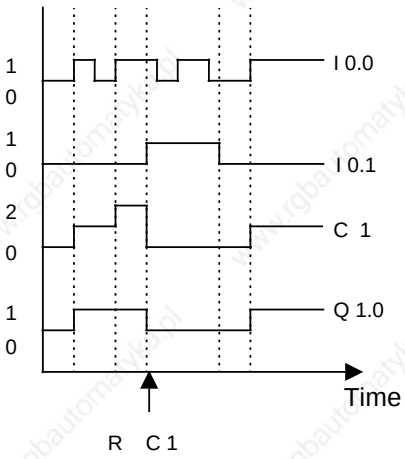


**Resetting a Counter “R” and Counting Up “CU”**

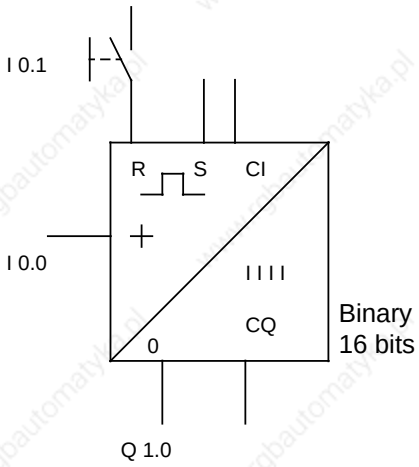
**Example:**

When input I 0.0 is switched on, the count in counter 1 is incremented by 1. As long as a second input (I 0.1) is “1”, the count is reset to “0”.
The A C 1 operation results in signal state “1” at output Q 1.0 as long as the count is not “0”.

**Timing Diagram**



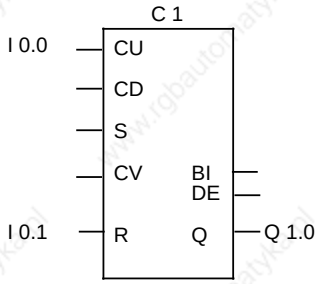

**Circuit Diagram**



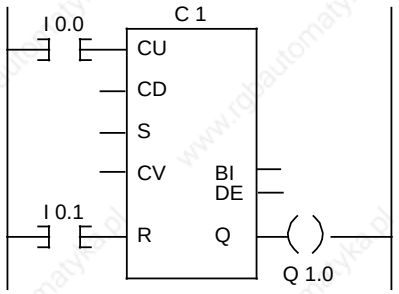

**STL**

```
A     I    0.0
CU    C    1
NOP   0
NOP   0
NOP   0
A     I    0.1
R     C    1
NOP   0
NOP   0
A     C    1
=     Q    1.0
```

**CSF**




**LAD**



### 8.1.6 Comparison Operations

Comparison operations compare the contents of the two accumulators. The comparison does not change the accumulators' contents. Table 8-6 provides an overview of the comparison operations. An example follows the table.

**Table 8-6. Overview of Comparison Operations**

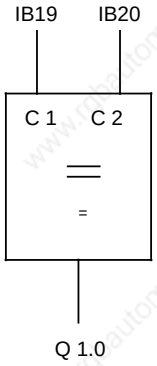
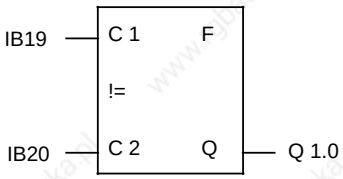
| Operation | Operand | | Meaning |
|---|---|---|---|
| **! = F** | | | **Compare for "equal to"**<br>The contents of the two accumulators are interpreted as bit patterns and scanned to see if they are equal. |
| **> < F** | | | **Compare for "not equal to"**<br>The contents of the two accumulators are interpreted as bit patterns and compared to see if they are not equal. |
| **> F** | | | **Compare for "greater than"**<br>The contents of the two accumulators are interpreted as fixed-point numbers. They are compared to see if the operand in ACCU 2 is greater than the operand in ACCU 1. |
| **> = F** | | | **Compare for "greater than or equal to"**<br>The contents of the two accumulators are interpreted as fixed-point numbers. They are compared to see if the operand in ACCU 2 is greater than or equal to the operand in ACCU 1. |
| **< F** | | | **Compare for "less than"**<br>The contents of the two accumulators are interpreted as fixed-point numbers. They are compared to see if the operand in ACCU 2 is less than the operand in ACCU 1. |
| **< = F** | | | **Compare for "less than or equal to"**<br>The contents of the two accumulators are interpreted as fixed-point numbers. They are compared to see if the operand in ACCU 2 is less than or equal to the operand in ACCU 1. |

#### Processing Comparison Operations

To compare two operands, load them consecutively into the two accumulators. Execution of the operations is independent of the RLO.
The result is binary and is available as RLO for further program processing. If the comparison is satisfied, the RLO is "1". Otherwise it is "0".
Executing the comparison operations sets the condition codes (see section 8.4).

> **Note**
>
> When using comparison operations, make sure the operands have the same number format.

**Example:** The values of input bytes IB19 and IB20 are compared. If they are equal, output Q 1.0 is set.

| Circuit Diagram | STL | CSF/LAD |
|---|---|---|
|  | L IB 19<br>L IB 20<br>!=F<br>= Q 1.0 |  |

## 8.1.7 Arithmetic Operations

Arithmetic operations interpret the contents of the accumulators as fixed-point numbers and manipulate them. The result is stored in ACCU 1. Table 8-7 provides an overview of the arithmetic operations. An example follows the table.

**Table 8-7. Overview of Arithmetic Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **+ F** | | | **Addition**<br>The contents of both accumulators are added. |
| **- F** | | | **Subtraction**<br>The contents of ACCU 1 are subtracted from the contents of ACCU 2. |

CPU 102 and higher have integral function blocks for multiplication and division (see section 9.2).

### Processing an Arithmetic Operation

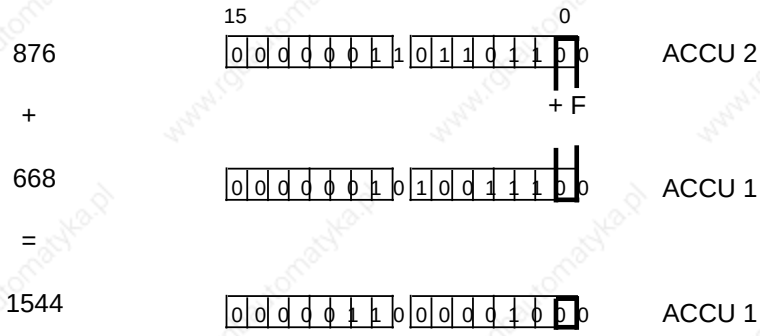
Before an arithmetic operation is executed, both operands must be loaded into the accumulators.

> **Note**
>
> When using arithmetic operations, make sure the operands have the same number format.

Arithmetic operations are executed independently of the RLO. The result is available in ACCU 1 for further processing. The contents of ACCU 2 are not changed.
These operations do not affect the RLO. The condition codes are set according to the results.

| STL | Explanation |
|---|---|
| `L C 3` | The value of counter 3 is loaded into ACCU 1. |
| `L C 1` | The value of counter 1 is loaded into ACCU 1. The previous contents of ACCU 1 are shifted to ACCU 2. |
| `+ F` | The contents of the two accumulators are interpreted as 16-bit fixed-point numbers and added. |
| `T QW12` | The result, contents of ACCU 1, is transferred to output word QW12. |

**Numeric Example**



## 8.1.8 Block Call Operations

Block call operations specify the sequence of a structured program. Table 8-8 provides an overview of the block call operations. Examples follow the table.

**Table 8-8. Overview of Block Call Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **JU** | | **Jump unconditionally**<br>Program scanning continues in a different block regardless of the RLO.<br>The RLO is not affected. |
| **JC** | | **Jump conditionally**<br>Program scanning jumps to a different block when the RLO is “1”.<br>Otherwise program scanning continues in the same block.<br>The RLO is set to “1”. |
| | **ID** / **Parameter** | **CPU 100** / **CPU 102** / **CPU 103**<br>OB: 0 to 63 / 0 to 63 / 0 to 255<br>PB: 0 to 63 / 0 to 63 / 0 to 255<br>FB: 0 to 63 / 0 to 63 / 0 to 255<br>SB: — / — / 0 to 255 |
| **C** | | **Call a data block**<br>A data block is activated regardless of the RLO.<br>Program scanning is not interrupted.<br>The RLO is not affected. |
| **G** | | **Generate and delete a data block***<br>An area is set up in the RAM to store data regardless of the RLO. |
| | **ID** / **Parameter** | **CPU 100** / **CPU 102** / **CPU 103**<br>DB: 2 to 63** / 2 to 63** / 2 to 255** |
| **BE** | | **Block end**<br>The current block is terminated regardless of the RLO.<br>Program scanning continues in the block in which the call originated.<br>The RLO is “carried along” but not affected. BE is always the last statement in a block. |
| **BEU** | | **Block end, unconditional**<br>The current block is terminated regardless of the RLO.<br>Program scanning continues in the block in which the call originated.<br>The RLO is “carried along” but not affected. |
| **BEC** | | **Block end, conditional**<br>When the RLO is “1”, the current block is terminated.<br>Program scanning continues in the block in which the call originated.<br>During the block change, the RLO remains “1”.<br>If the RLO is “0”, the operation is not executed.<br>The RLO is set to “1” and linear program scanning continues. |

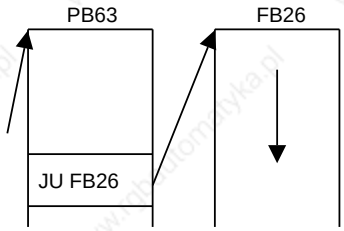
* The length of the DB must be loaded into ACCU 1 before execution of the operation. A length of 0 makes the DB invalid.

** Data blocks DB0 and DB1 are reserved for special functions.

### Unconditional Block Call “JU”

One block is called within another block, regardless of conditions.

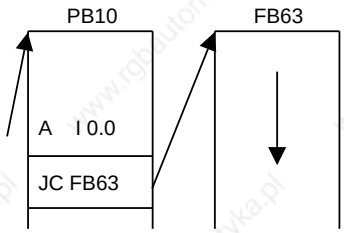
**Example:** A special function has been programmed in FB26. It is called at several locations in the program, e.g., in PB63, and processed.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | .<br>.<br>.<br>.<br>.<br>JU FB 26<br>. | The “JU FB26” statement in program block PB63 calls function block FB26. |

### Conditional Block Call “JC”

One block is called within another block when the previous condition has been satisfied (RLO = “1”).

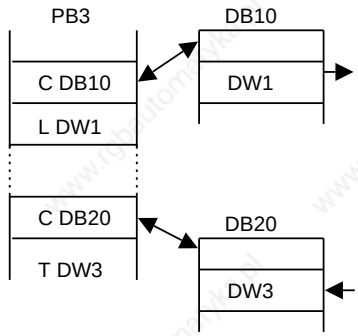
**Example:** A special function has been programmed in FB63. It is called and processed under certain conditions, e.g., in PB10.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | .<br>.<br>.<br>S F 1.0<br>A I 0.0<br>JC FB 63<br>. | The “JC FB63” statement in program block PB10 calls function block FB63 if input I 0.0 is “1”. |

### Call a Data Block “C DB”

Data blocks are always called unconditionally. All data processed following the call refers to the data block that has been called. This operation cannot generate new data blocks. Blocks that are called must be programmed or created before program scanning.

**Example:** Program block PB3 needs information that has been programmed as data word DW1 in data block DB10. Other data, e.g., the result of an arithmetic operation, is stored as data word DW3 in data block DB20.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | C DB 10<br>L DW 1<br>.<br>.<br>.<br>.<br>C DB 20<br>T DW 3 | The information from data word DW1 in data block DB10 is loaded into the accumulator. The contents of ACCU 1 are stored in data word DW3 of data block DB20. |

### Generating and Deleting a Data Block

The “G DB x” statement does not call a data block. Instead, it generates a new block. If you want to use the data in this data block, call it with the “C DB” statement.
Before the “G DB” statement, indicate in ACCU 1 the number of data words the block is to have (see the example below).

If you specify zero as the data block length, the data block in question is deleted, i.e., it is removed from the address list. It is considered nonexistent.

**Note**

The block is stored in memory and is designated as invalid until the programmable controller memory is compressed (see section 7.5.3).

If you try to set up a data block that already exists, the “G DB x” statement is not executed.

A data block can be a maximum of 256 data words (DW0 to 255) in length.

### Generating a Data Block

| Example | STL | Explanation |
|---|---|---|
| Generate a data block with 128 data words without the aid of a programmer. | L KF + 127<br>**G DB 5** | The constant fixed-point number +127 is loaded into ACCU 1. At the same time, the old contents of ACCU 1 are shifted to ACCU 2. Data block 5 is generated with a length of 128 data words (0000) in the RAM of the PLC and entered in the block address list. The next time the “G DB5” operation is processed, it has no effect if the contents of ACCU 1 are not 0. |

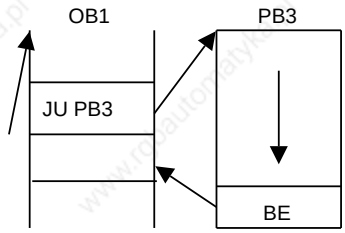
### Deleting a Data Block

| Example | STL | Explanation |
|---|---|---|
| Delete a data block that is no longer needed. | L KF + 0<br>G DB 5 | The constant fixed-point number +0 is loaded into ACCU 1. At the same time, the old contents of ACCU 1 are shifted to ACCU 2. Data block 5, which must be in the RAM of the PLC, is declared invalid and removed from the block address list. |

**Block End “BE”**

The “BE” operation terminates a block. Data blocks do not need to be terminated. “BE” is always the last statement in a block.
In structured programming, program scanning jumps back to the block where the call for the current block was made.
Boolean logic operations cannot be continued in a higher-order block.

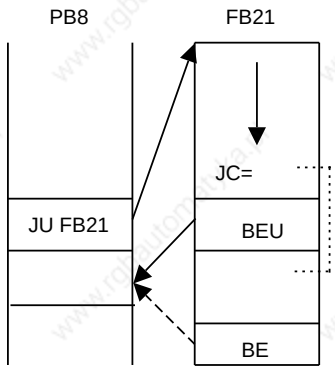
**Example:** Program block PB3 is terminated by the “BE” statement.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | .<br>.<br>.<br>.<br>.<br>.<br>BE | The “BE” statement terminates program block PB3 and causes program scanning to return to organization block OB1. |

**Unconditional Block End “BEU”**

The “BEU” operation causes a return within a block. However, jump operations can bypass the “BEU” operation in function blocks (see sections 8.2.10 and 8.3.4).
Binary logic operations cannot be continued in a higher-order block.

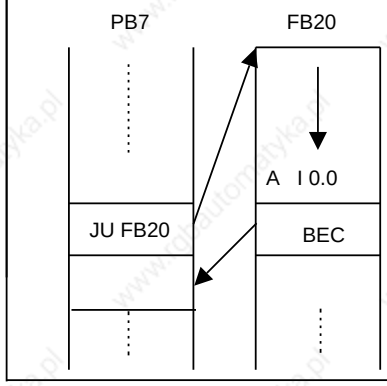
**Example:** Scanning of function block FB21 is terminated regardless of the RLO.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | .<br>.<br>.<br>.<br>JC=<br>BEU<br>.<br>.<br>.<br>.<br>BE | The “BEU” statement causes program scanning to leave function block FB21 and return to program block PB8. |

**Conditional Block End “BEC”**

The “BEC” operation causes a return within a block if the previous condition has been satisfied (RLO = 1).
Otherwise, linear program scanning is continued with RLO “1”.

**Example:** Scanning of program block FB20 is terminated if the RLO = “1”.

| Program Sequence | STL | Explanation |
|---|---|---|
|  | .<br>.<br>.<br>.<br>A I 0.0<br>BEC<br>.<br>.<br>.<br>. | The “BEC” statement causes program scanning to return to program block PB7 from function block FB20 if input I 0.0 is “1”. |

## 8.1.9 Other Operations

Table 8-9 lists other basic operations. Explanations follow the table.

**Table 8-9. Other Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **STP** | | | **Stop at the end of program scanning (in OB1)**<br>Current program scanning is terminated.<br>The PIQ is read out. Then the PLC goes into the STOP mode. |
| **NOP 0** | | | **“No” Operation**<br>Sixteen bits in the RAM are set to “0”. |
| **NOP 1** | | | **“No” Operation**<br>Sixteen bits in the RAM are set to “1”. |
| **BLD** | | | **Display Generation Operation**<br>“BLD” means a display generation operation for the programmer. |
| | **ID** | | **Parameter**<br>130, 131, 132, 133, 255 |

**Note**

These operations can be programmed in STL form only.

**STOP Operation**

The “STP” operation puts the programmable controller into the STOP mode. This can be desirable for time-critical system circumstances or when a programmable controller error occurs.

After the statement is processed, the control program is scanned to the end, regardless of the RLO. Afterwards the programmable controller goes into the STOP mode with the error ID “STS”. You can restart the programmable controller with the mode selector (STOP to RUN) or with a programmer.

**“NOP” (No Operations)**

The “NOP” operations reserve or overwrite memory locations.

**Display Generation Operations**

“BLD” display generation operations divide program parts into segments within a block.

“NOP” operations and display generation operations are significant only for the programmer when representing the STEP 5 program.
The programmable controller does not execute any operation when these statements are processed.

## 8.2 Supplementary Operations

Supplementary operations extend the operations set. However, compared to basic operations, which can be programmed in all blocks, supplementary operations have the following limitations.

- They can be programmed in function blocks only.
- They can be represented in STL form only.

The following sections describe the supplementary operations.

### 8.2.1 Load Operation, for CPU 103 and Higher

As with the basic load operations, the supplementary load operation copies information into the accumulator. Table 8-10 explains the load operation. An example follows the table.

**Table 8-10. Load Operation**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **L** | | | **Load**<br>A word from the system data is loaded into ACCU 1 regardless of the RLO. |
| | **ID**<br>RS | | **Parameter**<br>0 to 255 |

| Example | STL | Explanation |
|---|---|---|
| In order to set parameters for SINEC L1 bus operation via the system data, the programmer and slave numbers from SD57 should be input into ACCU 1. | . . .<br>L RS 57<br>. . . | Load ACCU 1 with the programmer and slave numbers. |

## 8.2.2 Enable Operation, for CPU 103 and Higher

You can use the enable operation (FR) to execute the following operations even without an edge change.

- Start a timer
- Set a counter
- Count up and down

Table 8-11 presents the enable operation. An example follows the table.

**Table 8-11. Enable Operation**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **FR** | | | **Enable a Timer/Counter**<br>Timers and counters are enabled on the leading edge of the RLO. This operation restarts a timer, sets a counter, or causes a counter to count up or down when the RLO “1” is pending at the “Start” operation. |
| | **ID** | **Parameter** | |
| | T | 0 to 127 | |
| | C | 0 to 127 | |

| Example | STL | Explanation |
|---|---|---|
| Input I 0.0 starts a timer T 2 as an extended pulse (pulse width 50 s). This timer sets output Q 1.0 for the duration of the pulse.<br>.<br>.<br>.<br>. | A I 0.0<br>L KT 500.1<br>SE T 2<br>A T 2<br>= Q 1.0<br>.<br>.<br>.<br>. | Start a timer T 2 as an extended pulse.<br>Output Q 1.0 is set for 50 s. |
| If output Q 1.1 is reset repeatedly, the timer should also be restarted repeatedly. | A Q 1.1<br>**FR T 2**<br><br>BE | If output Q 1.1 is set (positive edge change of the RLO) during the time in which input I 0.0 is set, timer T 2 is restarted. Output Q 1.0 therefore remains set at the restarted time or is reset.<br>If input I 0.0 is not set during the edge change of output Q 1.1, the timer is not restarted. |

### 8.2.3 Bit Test Operations, for CPU 103 and Higher

Bit test operations scan digital operands bit by bit and affect them. Bit test operations must always be at the beginning of a logic operation. Table 8-12 provides an overview of these operations.

**Table 8-12. Overview of Bit Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **TB** | | | **Test a bit for signal state "1"**<br>A single bit is scanned regardless of the RLO. The RLO is affected according to the bit's signal state (see Table 8-13). |
| **TBN** | | | **Test a bit for signal state "0"**<br>A single bit is scanned regardless of the RLO. The RLO is affected according to the bit's signal state (see Table 8-13). |
| **SU** | | | **Set a bit unconditionally**<br>The addressed bit is set to "1" regardless of the RLO. The RLO is not affected. |
| **RU** | | | **Reset a bit unconditionally**<br>The addressed bit is set to "0" regardless of the RLO. The RLO is not affected. |
| | **ID** | **Parameter** | |
| | T | 0.0 to 127.15 | |
| | C | 0.0 to 127.15 | |
| | D | 0.0 to 255.15 | |
| | RS[1] | 0.0 to 255.15 | |

[1] RS applies only to TB and TBN

Table 8-13 shows how the RLO is formed during the bit test operations "TB" and "TBN". An example for applying the bit operations follows the table.

**Table 8-13. Effect of "TB" and "TBN" on the RLO**

| Operation | TB | | TBN | |
|---|---|---|---|---|
| Signal state of the bit in the operand indicated | 0 | 1 | 0 | 1 |
| Result of logic operation | 0 | 1 | 1 | 0 |

| Example | STL | Explanation |
|---|---|---|
| A photoelectric barrier that counts piece goods is installed at input I 0.0. After every 100 pieces, the program is to jump to FB5 or FB6. After 800 pieces, counter 10 is to be reset automatically and start counting again. | C DB 10<br>A I 0.0<br>CU C 10<br>A I 0.1<br>L KC 000<br>S C 10<br>O I 0.2<br>O F 5.2<br>R C 10<br>LD C 10<br>T DW 12<br><br>**TBN D 12.8**<br><br>JC FB 5<br><br>**TB D 12.8**<br><br>JC FB 6<br><br>**TB D 12.11**<br><br>= F 5.2 | Call data block 10.<br><br>Input I 0.1 loads the count of counter 10 with the constant 0. With each positive edge change at I 0.0, the counter is incremented by 1. The counter is reset by either input I 0.2 or flag F 5.2.<br><br>The current count of the counter is stored in data word DW12 in BCD code.<br><br>As long as bit 8 of data word DW12 is zero, program processing jumps to function block FB5. This is the case for the first, third, fifth etc. batch of 100 pieces.<br>As long as bit 8 of data word DW12 is “1”, program scanning jumps to function block FB 6. This is the case for the second, fourth, sixth, etc. batch of 100 pieces.<br>When data bit 11 of data word DW12 becomes “1” (the count is then 800), flag F 5.2 is set conditionally. |
| A photoelectric barrier that counts piece goods is installed at input I 0.3. After every 256 pieces, the counter is supposed to be reset and start counting again. | :A I 0.3<br>:CU C 2<br>:A I 0.4<br>:L KC 000<br>:S C 20<br><br>**:TB C 20.8**<br><br>:JC = FULL<br>:BEU<br><br>**FULL:RU C 20.8**<br>:BE | Input I 0.4 loads the count of counter 20 with the constant 0. The count is incremented by 1 with each positive edge change at input I 0.3. If the count has reached 256 = $100_H$ (bit 8 is “1”), program scanning jumps to the label “FULL”. Otherwise the block is terminated.<br><br>Bit 8 of counter C 20 is set to “0” unconditionally. Then the count is again $000_H$. |

## Note

Times and counts are stored in the timer/counter word in hexadecimal notation in the 10 least significant bits (bits 0 to 9).
The time base is stored in bits 12 and 13 of the timer word.

### 8.2.4 Digital Logic Operations

Digital logic operations combine the contents of both accumulators logically bit by bit.
Table 8-14 provides an overview of these digital logic operations. Examples follow the table.

**Table 8-14. Overview of Digital Logic Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **AW** | | **Combine bit by bit through logic AND** |
| **OW** | | **Combine bit by bit through logic OR** |
| **XOW** | | **Combine bit by bit through logic EXCLUSIVE OR** |

#### Processing a Digital Logic Operation

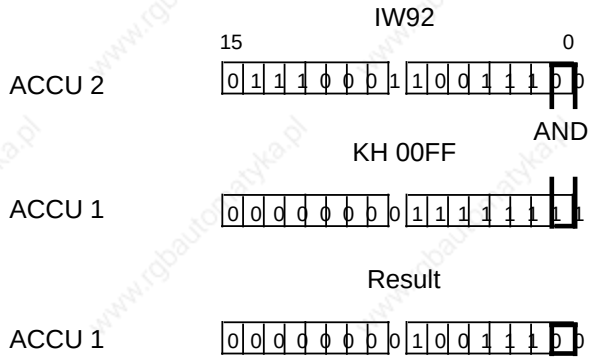
A digital logic operation is executed regardless of the RLO. It also does not affect the RLO.
However, it sets condition codes according to the result of the arithmetic operation (see section 8.4).

> **Note**
>
> Make sure both operands have the same number format. Then load them into the accumulators before executing the operation.

The result of the arithmetic operation is available in ACCU 1 for further processing. The contents of ACCU 2 are not affected.

| STL | | Explanation |
|---|---|---|
| L | IW 92 | Load input word IW92 into ACCU 1. |
| L | KH 00FF | Load a constant into ACCU 1. The previous contents of ACCU 1 are shifted to ACCU 2. |
| AW | | Combine the contents of both accumulators bit by bit through logic AND. |
| T | QW 82 | Transfer the resulting contents from ACCU 1 to output word QW82. |

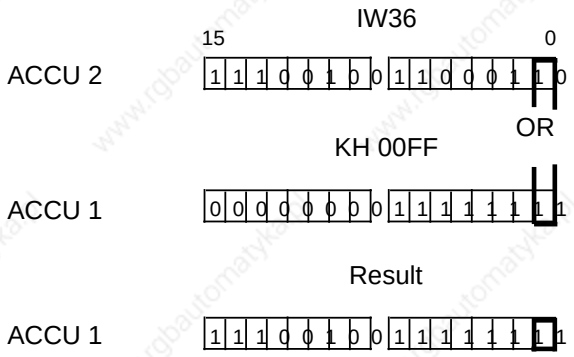
**Numeric Example**




Set the 8 high-order bits in input word IW92 to “0”.
Compare both words bit by bit. If corresponding bits are both “1”, the result bit is set to “1”.

| STL | | Explanation |
|---|---|---|
| L | IW 36 | Load input word IW36 into ACCU 1. |
| L | KH 00FF | Load a constant into ACCU 1. The previous contents of ACCU 1 are shifted to ACCU 2. |
| OW | | Combine the contents of both accumulators bit by bit through logic OR. |
| T | IW 36 | Transfer the result (contents of ACCU 1) to input word IW36. |

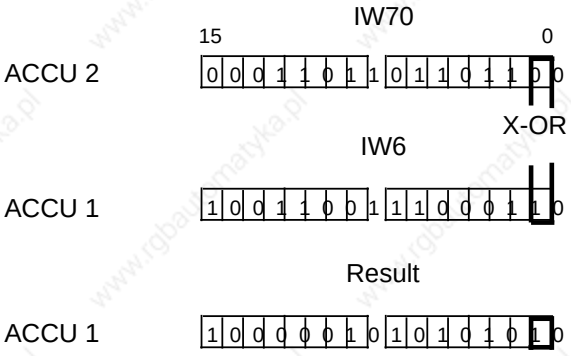
**Numeric Example**




Set the 8 low-order bits in input word IW36 to “1”. Compare both words bit by bit.
If either of the corresponding bits is “1”, a “1” is set in the result word.

| STL | | Explanation |
|---|---|---|
| L | IW 70 | Load input word IW70 into ACCU 1. |
| L | IW 6 | Load input word IW6 into ACCU 1. The previous contents of ACCU 1 are shifted to ACCU 2. |
| XOW | | Combine the contents of both accumulators bit by bit through logic EXCLUSIVE OR. |
| T | QW 86 | Transfer the result (contents of ACCU 1) to output word QW86. |

**Numeric Example**




Check to see if input words IW70 and IW6 are equal.
The result bit is set to “1” only if corresponding bits in ACCU 1 and ACCU 2 are unequal.

## 8.2.5 Shift Operations

Shift operations shift a bit pattern in ACCU 1. The contents of ACCU 2 are not affected. Shifting multiplies or divides the contents of ACCU 1 by powers of two. Table 8-15 provides an overview of the shift operations. Examples follow the table.

**Table 8-15. Overview of Shift Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **SLW** | | | **Shift to the left.**<br>The bit pattern in ACCU 1 is shifted to the left. |
| **SRW** | | | **Shift to the right.**<br>The bit pattern in ACCU 1 is shifted to the right. |
| | | **Parameter** | 0 to 15 |

### Processing a Shift Operation

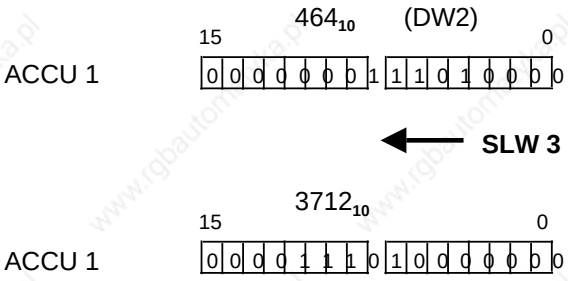
Execution of shift operations is unconditional. The RLO is not affected. However, shift operations set condition codes.
Consequently, the status of the last bit that is shifted out can be scanned with jump functions.

The shift statement parameter indicates the number of bit positions by which the contents of ACCU 1 are to be shifted to the left (SLW) or to the right (SRW). Bit positions vacated during shifting are assigned zeros.
The contents of the bits that are shifted out of ACCU 1 are lost. Following execution of the operation, the state of bit $2^0$ (SRW) or bit $2^{15}$ (SLW) has an influence on the CC1 bit, which can then be evaluated.

A shift operation with parameter “0” is handled like a “NOP” operation. The central processor processes the next STEP 5 statement with no further reaction.
Before executing a shift operation, load the operand to be processed into ACCU 1.
The altered operand is available there for further processing.

| STL | Explanation |
|---|---|
| L DW 2 | Load the contents of data word DW2 into ACCU 1. |
| SLW 3 | Shift the bit pattern in ACCU 1 three positions to the left. |
| T DW 3 | Transfer the result (contents of ACCU 1) to data word DW3. |

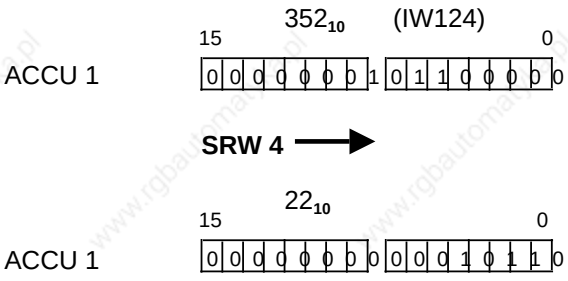
**Numeric Example**




The value $464_{10}$ is stored in data word DW2. Multiply this value by $2^3$=8. Do so by shifting the bit pattern of DW2 in ACCU 1 three positions to the left.

| STL | Explanation |
|---|---|
| L IW 124 | Load the value of input word IW124 into ACCU 1. |
| SRW 4 | Shift the bit pattern in ACCU 1 four positions to the right. |
| T QW 126 | Transfer the result (contents of ACCU 1) to output word QW126. |

**Numeric Example**




The value $352_{10}$ is stored in IW124. Shift the corresponding bit pattern in ACCU 1 four positions to the right to divide the value $352_{10}$ by $2^4$ = 16.

## 8.2.6 Conversion Operations

Conversion operations convert the values in ACCU 1. Table 8-16 provides an overview of the conversion operations. Examples follow the table.

**Table 8-16. Overview of Conversion Operations**

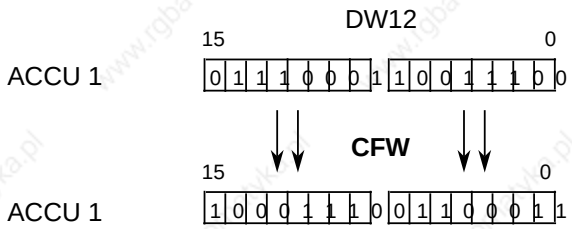
| Operation | Operand | Meaning |
|---|---|---|
| **CFW** | | **One's complement**<br>The contents of ACCU 1 are inverted bit by bit. |
| **CSW** | | **Two's complement**<br>The contents of ACCU 1 are inverted bit by bit. Afterwards the word $0001_H$ is added. |

### Processing Conversion Operations

Execution of these operations does not depend on the RLO nor does it affect the RLO.
The “CSW” operation sets the condition codes (see section 8.4).

| STL | Explanation |
|---|---|
| `L DW 12` | Load the contents of data word DW12 into ACCU 1. |
| `CFW` | Invert all bits in ACCU 1. |
| `T QW 20` | Transfer the new contents of ACCU 1 to output word QW20. |

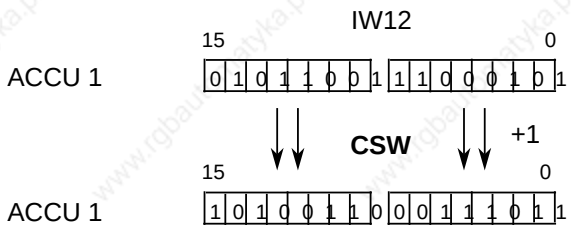
**Numeric Example**




In a system, normally open contacts have been replaced by normally closed contacts. If the information in data word DW12 is to maintain its previous effect, DW12 must be inverted.

| STL | Explanation |
|---|---|
| L IW 12 | Load the contents of input word IW12 into ACCU 1. |
| CSW | Invert all bits and add a “1”. |
| T DW 100 | Transfer the altered word to data word DW100. |

**Numeric Example**




Form the negative value of the value in input word IW12.

## 8.2.7 Decrement/Increment, for CPU 103 and Higher

The decrement/increment operations change the data loaded into ACCU 1. Table 8-17 provides an overview of the decrement/increment operations. An example follows the table.

**Table 8-17. Decrement/Increment Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **D** | | | **Decrement**<br>Decrement the contents of the accumulator. |
| **I** | | | **Increment**<br>Increment the contents of the accumulator.<br>The contents of ACCU 1 are either decremented or incremented by the number indicated in the parameter.<br>Execution of the operation is unconditional and is limited to the right-hand byte (without carry). |
| | | | **Parameter**<br>0 to 255 |

### Processing

Execution of the decrement and increment operations is independent of the RLO and does not affect the RLO or the condition codes.
The parameter indicates the value by which the contents of ACCU 1 are to be changed.
The operations refer to decimal values; however, the result is stored in ACCU 1 in binary form.
Changes relate only to the low byte in the accumulator.

| Example | STL | Explanation |
|---|---|---|
| Increment the hexadecimal constant $1010_H$ by 16 and store the result in data word DW8. | C DB 6<br>L KH 1010<br>I 16 | Call data block DB6.<br>Load hexadecimal constant $1010_H$ into ACCU 1.<br>Increment the low byte of ACCU 1 by 16. The result, $1020_H$, is located in ACCU 1. |
| In addition, decrement the incrementation result by 33 and store the new result in data word DW9. | T DW 8<br><br>D 33<br><br>T DW 9 | Transfer the contents of ACCU 1 ($1020_H$) to data word DW8. Since the incrementation result is still in ACCU 1, you can decrement by 33 directly.<br>The result would be $FFF_H$. However, since the high byte of ACCU 1 is not decremented along with the low byte, the result in ACCU 1 is $10FF_H$.<br>The contents of ACCU 1 are transferred to DW9 ($10FF_H$). |

## 8.2.8 Disable/Enable Interrupt, for CPU 103 Version 8MA02 and Higher

The disable/enable interrupt operations affect interrupt-driven and time-controlled program scanning. They prevent process or time interrupts from interfering with the processing of a sequence of statements or blocks. Table 8-18 lists the disable/enable interrupt operations. An example follows the table.

**Table 8-18. Disable/Enable Interrupt Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **IA** | | | **Disable interrupt** |
| **RA** | | | **Enable interrupt** |

**Processing**

Execution of the disable/enable interrupt operations does not depend on the RLO. These operations do not affect the RLO or the condition codes. After the “IA” statement is processed, no more interrupts are executed. The “RA” statement cancels the effect of “IA”.

| Example | STL | Explanation |
|---|---|---|
| Disable interrupt processing in a specific program section and then enable it again. | . | |
| | . | |
| | . | |
| | . | |
| | . | |
| | = Q 1.0 | |
| | **IA** | Disable interrupt. |
| | A I 0.0 | |
| | . | |
| | . | |
| | . | |
| | JU FB 3 | If an interrupt occurs, the program section between the “IA” and “RA” is scanned without interruption. |
| | . | |
| | . | |
| | . | |
| | **RA** | Enable interrupt. Interrupts that occurred in the meantime are processed after the “RA” operation. |
| | . | |
| | . | |
| | . | |

## 8.2.9 “DO” Operation, for CPU 103 and Higher

Use the “DO” operation to process STEP 5 statements as indexed operations. This allows you to change the parameter of an operand during control program processing (see Table 8-19).

**Table 8-19. Overview of the “DO” Operation**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **DO** | | | **Processing a flag word or data word** |
| | **ID** ↑<br>FW<br>DW | ↑<br>**Parameter**<br>0 to 254<br>0 to 255 | |

**“DO” Statements**

“DO flag word or data word x” is a two-word statement that is unaffected by the RLO. “DO” consists of the following two statements:

- The first statement contains the “DO” operation and a flag word or data word.
- The second statement defines the operation and the operand identifier you want the control program to process. You must enter 0 or 0.0 as the parameter.

The control program works with the parameter that is stored in the flag word or data word. This parameter is the one called up in the first statement. If you want to index binary operations, inputs, outputs, or flags, you input the bit address in the high byte of this word. You input the byte address in the low byte. In any other instance, the high byte must be “0”.

You can combine the following operations with the “DO” statement:

| Operations | Explanations |
|---|---|
| A[1], AN, O, ON | Boolean logic operations |
| S, R, = | Set/reset operations |
| FR T, RT, SF T, SD T, SP T, SS T, SE T, | Timer operations |
| FR C, RC, SC, CD C, CU C | Counter operations |
| L, LD, T | Load and transfer operations |
| JU=, JC=JZ=, JN=, JP=, JM=, JO= | Jump operations |
| SLW, SRW | Shift operations |
| D, I | Decrement and increment |
| C DB, JU, JC, TNB | Block calls |

[1] In combination with “DO FW,” the “A I” operation becomes the “A Q” operation if the byte address in the data word or flag word is higher than 127.

**Caution**

Damage to the system.
Performing operations that are not listed in Table 8-20 will damage your system.
Perform only those operations that are listed in Table 8-20.

Figure 8-6 shows how the contents of a data word determine the parameter of the next statement.

| | DB6 |
|---|---|
| DW 12 | KH = **0108** |
| DW 13 | KH = 0001 |

FBx

```
:C   DB   6
     .
     .
     .
:DO  DW   12
:A   I    0.0
:DO  DW   13
:FR  T    0
```

Actual program

```
:C   DB   6
     .
     .
     .
     .
:A   I    8.1
     .
:FR  T    1
```

**Figure 8-6. Executing a "DO" Operation**

The following example illustrates how new parameters are generated in every program scan.

| Example | STL | Explanation |
|---|---|---|
| Set the contents of data words DW20 to DW100 to signal state "0". The index register for the parameter for the data words is DW1. | `:C DB 202` | Call data block DB202. |
| | `:L KB 20` | Load constant number 20 in ACCU 1. |
| | `:T DW 1` | Transfer contents from ACCU 1 to data word DW1. |
| | `F 1 :L KH 0` | Load hex constant 0 in ACCU 1. |
| | **`:DO DW 1`** | DO data word DW1. |
| | **`:T DW 0`** | Transfer the contents from ACCU 1 to the data word whose address is stored in data word DW1. |
| | `:L DW 1` | Load data word DW1 in ACCU 1. |
| | `:L KB 1` | Load constant number 1 in ACCU 1. Data word DW1 is shifted to ACCU 2. |
| | `:+F` | ACCU 2 und ACCU 1 are added, and the result is stored in ACCU 1 (data word address is higher). |
| | `:T DW 1` | Transfer contents of ACCU 1 to data word DW1 (new data word address). |
| | `:L KB 100` | The constant number 100 is loaded in ACCU 1 and the new data word address is shifted to ACCU 2. |
| | `:<=F` | Compare the ACCUs for less than or equal to: ACCU 2 ACCU 1. |
| | `:JC = F 1` | Jump conditionally to label F1, if ACCU 2 ACCU 1. |

## 8.2.10 Jump Operations

Table 8.20 provides an overview of the jump operations. An example follows the table.

**Table 8-20. Overview of Jump Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **JU =** | | | **Jump unconditionally**<br>The unconditional jump is executed independently of conditions. |
| **JC=** | | | **Jump conditionally**<br>The conditional jump is executed if the RLO is “1”. If the RLO is “0”, the statement is not executed and the RLO is set to “1”. |
| **JZ =** | | | **Jump if the result is “zero”**<br>The jump is executed only if CC 1 = 0 and CC 0 = 0<br>The RLO is not changed. |
| **JN =** | | | **Jump if the result is “not zero”**<br>The jump is executed only if CC 1 CC 0<br>The RLO is not changed. |
| **JP =** | | | **Jump if the result is positive**<br>The jump is executed only if CC 1 = 1 and CC 0 = 0<br>The RLO is not changed. |
| **JM =** | | | **Jump if the result is negative**<br>The jump is executed only if CC 1 = 0 and CC 0 = 1<br>The RLO is not changed. |
| **JO =** | | | **Jump on overflow**<br>The jump is executed if an overflow occurs. Otherwise the jump is not executed. The RLO is not changed. |
| | **ID**<br>Jump label (up to 4 characters) | | |

### Processing Jump Operations

A symbolic jump destination (jump label) must always be entered next to a jump operation. This jump label can have up to four characters. The first character must be a letter of the alphabet.

When programming, please be aware of the following items:

- The absolute jump displacement cannot exceed +127 or - 128 words in the program memory. Some statements take up two words (e.g., “Load a constant”). For long jumps, insert an intermediate destination.
- Jumps can be executed only within a block.
- Jumping over segment boundaries (“BLD 255”) is not permitted.
- Jump labels can be set only at the start of a series of scans for CPU 102.

| Example | STL | Explanation |
|---|---|---|
| If no bit of input word IW1 is set, program scanning jumps to the label “AN 1”. If input word IW1 and output word QW3 do not agree, program processing jumps back to the label “AN 0”. Otherwise input word IW1 and data word DW12 are compared. If input word IW1 is greater than or less than data word DW12, program scanning jumps to the “DEST” label. | AN0 :L IW 1<br>:L KH 0000<br>:+F<br>**:JZ= AN 1**<br>:A I 0.0<br>.<br>.<br>.<br>.<br>.<br>.<br>AN1 :L IW 1<br>:L QW 3<br>:XOW<br>**:JN = AN 0**<br>:L IW 1<br>:L DW12<br>:>< F<br>**:JC = DEST**<br>.<br>.<br>.<br>.<br>.<br>DEST :A I 0.1<br>.<br>. | Load input word IW1 into ACCU 1. If the contents of ACCU 1 equal zero[1], jump to the label “AN 1”. Otherwise process the next statement (I 0.0).<br><br>Compare input word IW1 and output word QW3. If they are not equal, set individual bits in ACCU 1.<br>If the contents of ACCU 1 are not zero, jump to the label “AN 0”. Otherwise process the next statements.<br>Compare input word IW1 and data word DW12. If they are not equal, set RLO to “1”.<br>If the RLO = “1”, jump to the “DEST” label. If the RLO = “0”, process the next statement. |

1 The “L...” statement does not affect the condition codes. An addition (+F) is executed with the constant $0000_H$ so that the “JZ” operation can evaluate the contents of the accumulator.

## 8.2.11 Substitution Operations, for CPU 103 and Higher

If you plan to process a program with various operands and without a lot of changes, it is advisable to assign parameters to individual operands (see section 7.3.4). If you have to change the operands, you only need to reassign the parameters in the function block call.

These parameters are processed in the program as “formal operands”.
Special operations are necessary for this processing. However, these special operations are no different in their effect than operations without substitution. A brief description of these operations and examples follows.

**Binary Logic Operations**

Table 8-21 provides an overview of binary logic operations.

**Table 8-21. Overview of Binary Logic Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **A =** | | **AND operation**<br>Scan a formal operand for “1”. |
| **AN =** | | **AND operation**<br>Scan a formal operand for “0”. |
| **O =** | | **OR operation**<br>Scan a formal operand for “1”. |
| **ON =** | | **OR operation**<br>Scan a formal operand for “0”. |

| Formal operand | Actual operands permitted | Parameter type | Data type |
|---|---|---|---|
| | Inputs, outputs, and flags addressed in binary form | I, Q, F | BI |
| | Timers and counters | T, C | |

### Set/Reset Operations

Table 8-22 provides an overview of the set/reset operations. An example follows the table.

**Table 8-22. Overview of Set/Reset Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **S =** | | **Set a formal operand (binary).** |
| **RB =** | | **Reset a formal operand (binary).** |
| **= =** | | **Assign**<br>The RLO is assigned to a formal operand. |

| Formal operand | Actual operands permitted | Parameter type | Data type |
|---|---|---|---|
| | Inputs, outputs, and flags addressed in binary form | I, Q, F | BI |

**Example:** FB30 is assigned parameters in OB1.

| Call in OB1 | Program in FB30 | Executed Program |
|---|---|---|
| :JU FB 30 | **:A =ON 1** | :A I 0.0 |
| NAME :COMBINE | **:AN =ON 2** | :AN I 0.1 |
| ON 1 : I 0.0 | **:O =ON 3** | :O I 0.2 |
| ON 2 : I 0.1 | **:S =MOT 5** | :S Q 1.2 |
| ON 3 : I 0.2 | **:= =OFF 1** | := Q 1.0 |
| VAL1 : I 0.3 | **:A =VAL 1** | :A I 0.3 |
| OFF1 : Q 1.0 | **:A =ON 2** | :A I 0.1 |
| OFF2 : Q 1.1 | **:ON =ON 3** | :ON I 0.2 |
| MOT5 : Q 1.2 | **:RB =MOT 5** | :R Q 1.2 |
| : BE | **:= =OFF 2** | := Q 1.1 |
| | **:BE** | :BE |

### Load and Transfer Operations

Table 8-23 lists the various load and transfer operations. An example follows the table.

**Table 8-23. Overview of Load and Transfer Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **L =** | | **Load a formal operand.** |
| **LD =** | | **Load a formal operand in BCD code.** |
| **LW =** | | **Load the bit pattern of a formal operand.** |
| **T =** | | **Transfer to a formal operand.** |

| Formal operand | Actual operands permitted | Parameter type | Data type |
|---|---|---|---|
| For L = | Inputs, outputs, and flags addressed in binary form<br>Data<br>Timers and counters | I, Q, F<br>PW*, PY*<br>DW, DR, DL<br>T, C | BY, W |
| For LD = | Timers and counters | T, C | |
| For LW = | Bit pattern | D | KF, KH, KM, KY, KS, KT, KC |
| For T = | Inputs, outputs, data (DW, DR, DL) and flags addressed in binary form | I, Q<br>DW, DR, DL<br>F, PW*, PY* | BY, W |

* Not for integral function blocks

**Example:** FB34 is assigned parameters in PB1.

**Call in PB1**

```
        :JU FB 34
NAME    :LOAD/TRAN
I0      :   I 0.0
I1      :   I 0.1
L1      :   FW 10
LW1 :   :   KC 140
LC1     :   C 7
T1      :   QW 4
LW2 :   :   KC 160
        :BE
```

**Program in FB34**

```
:A      =I 0
:L      =L1
:S      C       6
:A      =I 1
:LW     =LW1
:S      C       7
:A      I       0.2
:CU     C       6
:CU     C       7
:LD     =LC1
:T      =T1
:A      I       0.3
:R      C       6
:R      C       7
:LW     =LW2
:LD     =LC1
:!=F
:R      C       7
:BE
```

**Executed Program**

```
:A      I       0.0
:L      FW      10
:S      C       6
:A      I       0.1
:L      KC      140
:S      C       7
:A      I       0.2
:CU     C       6
:CU     C       7
:LD     C       7
:T      QW      4
:A      I       0.3
:R      C       6
:R      C       7
:L      KC      160
:LD     C       7
:!=F
:R      C       7
:BE
```

### Timer and Counter Operations

Table 8-24 provides an overview of timer and counter operations. Examples follow the table.

**Table 8-24. Overview of Timer and Counter Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **FR =** | | **Enable** a formal operand for a cold restart. (For a description, see “FT” or “FC”, according to the formal operand). |
| **RD =** | | **Reset** a formal operand (digital). |
| **SP =** | | **Start** a pulse timer specified as a formal operand using the value stored in the accumulator. |
| **SD =** | | **Start** an on-delay timer specified as a formal operand using the value stored in the accumulator. |
| **SEC =** | | **Start** an extended pulse timer specified as a formal operand using the value stored in the accumulator or **set** a counter specified as a formal operand using the count specified in the accumulator. |
| **SSU =** | | **Start** a stored on-delay timer specified as a formal operand using the value stored in the accumulator or start the **count up** of a counter specified as a formal operand. |
| **SFD =** | | **Start** an off-delay timer specified as a formal operand using the value stored in the accumulator or start the **count down** of a counter specified as a formal operand. |

| Formal operand | Actual operands permitted | Parameter type | Data type |
|---|---|---|---|
| | Timers and counters[1] | T, C[1] | |

[1] “SP” and “SD” do not apply to counters.

### Specifying Times and Counts

As with the basic operations, you can specify a time or count as a formal operand. In this case, you must distinguish as follows whether the value is located in an operand word or is specified as a constant.

- Operand words can be of parameter type “I” or “Q” and of data type “W”. Use the “L=” operation to load them into the accumulator.
- Constants can be of parameter type “D” and of data type “KT” or “KC”. Use “LW=” to load these formal operands into the accumulator.

The following examples show how to work with timer and counter operations:

**Example 1:**

| Function Block Call | Program in Function Block (FB32) | Executed Program |
|---|---|---|
| :JU FB 32<br>NAME :TIME<br>I 5 : I 0.0<br>I 6 : I 0.1<br>TIM5 : T 5<br>TIM6 : T 6<br>OFF6 : Q 1.0<br>:BE | :AN =I 5<br>:A =I 6<br>:L KT 005.2<br>**:SFD =TIM5**<br>:A =I 5<br>:AN =I 6<br>:L KT 005.2<br>**:SSU =TIM6**<br>:A =TIM5<br>:O =TIM6<br>:= =OFF6<br>:A I 0.2<br>**:RD =TIM5**<br>**:RD =TIM6**<br>**:BE** | :AN I 0.0<br>:A I 0.1<br>:L KT 5.2<br>:SF T 5<br>:A I 0.0<br>:AN I 0.1<br>:L KT 5.2<br>:SS T 6<br>:A T 5<br>:O T 6<br>:= Q 1.0<br>:A I 0.2<br>:R T 5<br>:R T 6<br>:BE |

**Example 2:**

| Function Block Call | Program in Function Block (FB33) | Executed Program |
|---|---|---|
| :JU FB 33<br>NAME :COUNT<br>I2 : I 0.0<br>I3 : I 0.1<br>I4 : I 0.2<br>CNT5 : C 5<br>OFF3 : Q 1.0<br>:BE | :A =I 2<br>:L KC 017<br>**:SEC =CNT5**<br>:A =I 3<br>**:SSU =CNT5**<br>:A =I 4<br>**:SFD =CNT5**<br>:A =CNT5<br>:= =OFF3<br>:A I 0.3<br>**:RD =CNT5**<br>:BE | :A I 0.0<br>:L KC 017<br>:S C 5<br>:A I 0.1<br>:CU C 5<br>:A I 0.2<br>:CD C 5<br>:A C 5<br>:= Q 1.0<br>:A I 0.3<br>:R C 5<br>:BE |

### "DO" Operation

Table 8-25 and the example that follows explain the processing operation.

**Table 8-25. "DO" Operation**

| Operation | Operand | Meaning | | |
|---|---|---|---|---|
| **DO =** | | **Process formal operand**<br>The substituted blocks are called unconditionally. | | |
| **Formal operands** | | **Actual operands permitted** | **Parameter type** | **Data type** |
| | | DB, PB, SB, FB[1] | B | |

[1] As actual operands, function blocks cannot have block parameters.

**Example:**

| Function Block Call | Program in Function Block FB35 | Executed Program |
|---|---|---|
| STL<br>:JU FB 35<br>NAME :DO<br>D5 : DB 5<br>DW2 : DW 2<br>D6 : DB 6<br>DW1 : DW 1<br>Q4 : QW 4<br>MOT5 : FB 36<br>:BE | **:DO =D5**<br>:L =DW2<br>**:DO =D6**<br>:T =DW1<br>:T =Q4<br>**:DO =MOT5**<br>:BE | :C DB 5<br>:L DW 2<br>:C DB 6<br>:T DW 1<br>:T QW 4<br>:JU FB 36<br>:BE |

## 8.3 System Operations, for CPU 103 and Higher

System operations and supplementary operations have the following limitations:

- You can program them only in function blocks.
- You can program them only in the STL method of representation.

Since system operations access system data, only users with system knowledge should use them. If you want to program system operations, you must select “SYS: OPS. Y” in the programmer presets menu.

### 8.3.1 Set Operations

Like the supplementary bit operations, these set operations can change individual bits. Table 8-26 provides an overview of the set operations.

**Table 8-26. Overview of Set Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **SU** | | | **Set bit unconditionally**<br>A specific bit is set to “1” in the system data area. |
| **RU** | | | **Reset bit unconditionally**<br>A specific bit is set to “0” in the system data area. |
| | **ID**<br>RS | | **Parameter**<br>0.0 to 255.15 |

**Processing Set Operations**

Execution of set operations does not depend on the RLO.

### 8.3.2 Load and Transfer Operations

Use these load and transfer operations to address the entire program memory of the programmable controller. They are used mainly for data exchange between the accumulator and memory locations that cannot be addressed by operands. Table 8-27 provides an overview of the load and transfer operations.

**Table 8-27. Overview of Load and Transfer Operations**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **LIR** | | | **Load the register indirectly**<br>The contents of a memory word are loaded into the specified register (ACCU 1, 2). The address is in ACCU 1. |
| **TIR** | | | **Transfer the register indirectly**<br>The contents of the indicated register are transferred to a memory location. The address is in ACCU 1. |
| | | **Parameter**<br>0 (for ACCU 1), 2 (for ACCU 2) | |
| **TNB** | | | **Transfer a data field (byte-by-byte)**<br>A memory area is transferred in the program memory as a field.<br>End address destination area: ACCU 1<br>End address source area: ACCU 2 |
| **T** | | | **Transfer**<br>A word is transferred to the system data area. |
| | **ID**<br>RS | **Parameter**<br>0 to 255 | |

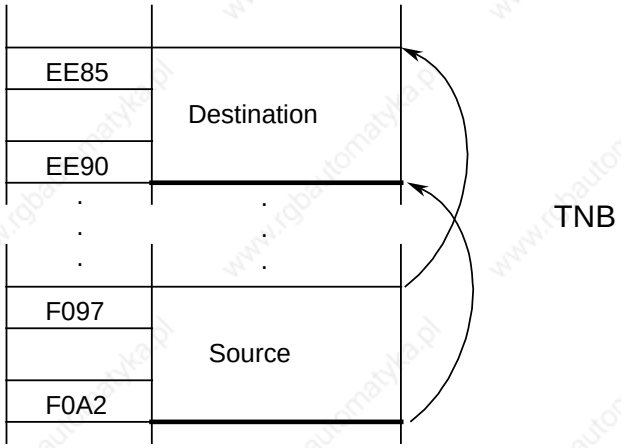
### Loading and Transferring Register Contents

Both accumulators can be addressed as registers. Each register is 16 bits wide. Since the “LIR” and “TIR” operations transmit data by words, the registers are addressed in pairs.
Loading and transferring register contents are independent of the RLO. The processor goes to ACCU 1 to get the address of the memory location referenced during data exchange. Consequently, make sure that the desired address is stored in ACCU 1 before this system operation is processed.

| STL | Explanation |
|---|---|
| .<br>. | |
| `L KH 6100` | Load the address $6100_H$ into ACCU 1. |
| `LIR 0` | Load the information from the memory location with the address $6100_H$ into ACCU 1. |

**Processing a Field Transfer**

A field transfer is processed independently of the RLO. The parameter indicates the length of the data field (in bytes) that is to be transferred. The field can be up to 255 bytes long.
The address of the source field is in ACCU 2. The address of the destination field is in ACCU 1. The higher address of each field must be specified because a field transfer takes place by decrementing. The bytes in the destination field are overwritten during the transfer.

| Example | Representation |
|---|---|
| Transfer a 12-byte data field from address $F0A2_H$ to address $EE90_H$. |  |



| STL | | | Explanation |
|---|---|---|---|
| :L | KH | F0A2 | Load the end address of the source field into ACCU 1. |
| :L | KH | EE90 | Load the end adress of the destination field into ACCU 1. The source address is shifted to ACCU 2. |
| :TNB | 12 | | Transfer the data field to the destination field. |

**Transferring to the System Data Area**

**Example:** Set the scan monitoring time to 100 ms after each mode change from “STOP” to “RUN”. You can program this time in multiples of 10 ms in system data word 96. The following function block can be called from OB21, for example.

| STL | Explanation |
|---|---|
| FB 11 | Block number and type |
| L KF +10 | Load ACCU 1 with the factor 10. |
| T RS 96 | Transfer this value to system data word 96. |
| BE | |

**Caution**

The TIR, TRS and TNB operations are memory-changing operations with which you can access the user memory and the system data area. These accesses are not monitored by the operating system. Improper use of the operations can lead to changes in the program and to a programmable controller crash.

## 8.3.3 Arithmetic Operations

An arithmetic operation changes the contents of ACCU 1 by a specified value. The parameter represents this value as a positive or negative decimal number. Table 8-28 shows the essential features of the “ADD” operation. An example follows the table.

**Table 8-28. Overview of the “ADD” Operation**

| Operation | Operand | | Meaning |
|---|---|---|---|
| **ADD** | | | **Add a constant**<br>Add byte or word constants. |
| | **ID** | **Parameter** | |
| | BF | -128 to +127 | |
| | KF | -32768 to +32767 | |

**Processing**

An arithmetic operation is executed independently of the RLO. It does not affect the RLO or the condition codes.
You can subtract by entering a negative parameter.
Even if the result cannot be represented by 16 bits, no carry is made to ACCU 2, i.e., the contents of ACCU 2 are not changed.

| Example | STL | | | Explanation |
|---|---|---|---|---|
| Decrement the constant $1020_H$ by 33 and store the result in flag word FW28. Afterwards add the constant 256 to the result and store the sum in flag word FW30. | L | KH | 1020 | The constant $1020_H$ is loaded into ACCU 1. |
| | ADD | BF | -33 | The constant $-33_{0D}$ is added to the ACCU contents. |
| | T | FW | 28 | The new ACCU contents ($0FFF_H$) are stored in flag word FW28. |
| | ADD | KF | 256 | The constant $256_{0D}$ is added to the last result. |
| | T | FW | 30 | The new ACCU contents ($10FF_H$) are stored in flag word FW30. |

## 8.3.4 Other Operations

Table 8-29 provides an overview of the remaining system operations.

**Table 8-29. The “TAK” and “STS” Operations**

| Operation | Operand | Meaning |
|---|---|---|
| **TAK** | | **Swap accumulator contents**<br>Swap the contents of ACCU 1 and ACCU 2 regardless of the RLO. The RLO and the condition codes are not affected. |
| **STS** | | **Stop immediately**<br>The PLC goes into the STOP mode regardless of the RLO. |

**Processing the “STS” Operation**

When the “STS” operation is executed, the programmable controller goes into the STOP mode immediately. Program processing is terminated at this point. The STOP state can only be cancelled manually (with the mode selector) or with the programmer function “PC START”.

## 8.4 Condition Code Generation

The processor of the programmable controller has the following three condition codes:
- CC 0
- CC 1
- OV (overflow)

The following operations affect the condition codes.
- Comparison operations
- Arithmetic operations
- Shift operations
- Some conversion operations

The state of the condition codes represents a condition for the various jump operations.

### Condition Code Generation for Comparison Operations

Execution of comparison operations sets condition codes CC 0 and CC 1 (see Table 8-30). The overflow condition code is not affected. Comparison operations do affect the RLO. When a comparison is satisfied, the RLO is 1. This allows you to use the “JC” conditional jump operation after a comparison operation.

**Table 8-30. Condition Code Settings for Comparison Operations**

| Contents of ACCU 2 as Compared to Contents of ACCU 1 | Condition Codes | | | Possible Jump Operations |
|---|---|---|---|---|
| | CC 1 | CC 0 | OV | |
| Equal to | 0 | 0 | | JZ |
| Less than | 0 | 1 | | JN, JM |
| Greater than | 1 | 0 | | JN, JP |

### Condition Code Generation for Arithmetic Operations

Execution of arithmetic operations sets all condition codes according to the result of the arithmetic operation (see Table 8-31).

**Table 8-31. Condition Code Settings for Fixed-Point Arithmetic Operations**

| Result after Arithmetic Operation is Executed | Condition Codes | | | Possible Jump Operations |
|---|---|---|---|---|
| | CC 1 | CC 0 | OV | |
| < - 32768 | 1 | 0 | 1 | JN, JP, JO |
| - 32768 to - 1 | 0 | 1 | 0 | JN, JM |
| 0 | 0 | 0 | 0 | JZ |
| +1 to +32767 | 1 | 0 | 0 | JN, JP |
| > +32767 | 0 | 1 | 1 | JN, JM, JO |
| (-) 65536* | 0 | 0 | 1 | JZ, JO |

* This number is the result of the calculation -32768 - 32768

### Condition Code Generation for Digital Logic Operations

Digital logic operations set CC 0 and CC 1. They do not affect the overflow condition code (see Table 8-32). The setting depends on the contents of the ACCU after the operation has been processed.

**Table 8-32. Condition Code Settings for Digital Logic Operations**

| Contents of the ACCU | Condition Codes | | | Possible Jump Operations |
|---|---|---|---|---|
| | CC 1 | CC 0 | OV | |
| Zero (KH = 0000) | 0 | 0 | | JZ |
| Not zero | 1 | 0 | | JN, JP |

### Condition Code Generation for Shift Operations

Execution of shift operations sets CC 0 and CC 1. It does not affect the overflow condition code (see Table 8-33). Code setting depends on the state of the last bit shifted out.

**Table 8-33. Condition Code Settings for Shift Operations**

| Value of the Last Bit Shifted Out | Condition Codes | | | Possible Jump Operations |
|---|---|---|---|---|
| | CC 1 | CC 0 | OV | |
| “0” | 0 | 0 | | JZ |
| “1” | 1 | 0 | | JN, JP |

### Condition Code Generation for Conversion Operations

The formation of the two's complement (CSW) sets all condition codes (see Table 8-34). The state of the condition codes is based on the result of the conversion function.

**Table 8-34. Condition Code Settings for Conversion Operations**

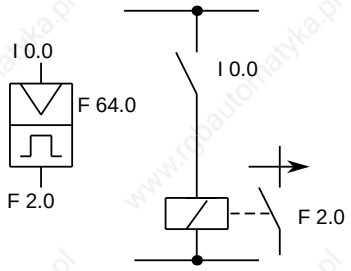
| Result after Arithmetic Operation is Executed | Condition Codes | | | Possible Jump Operations |
|---|---|---|---|---|
| | CC 1 | CC 0 | OV | |
| - 32768 * | 0 | 1 | 1 | JN, JM, JO |
| - 32767 to - 1 | 0 | 1 | 0 | JN, JM |
| 0 | 0 | 0 | 0 | JZ |
| +1 to +32767 | 1 | 0 | 0 | JN, JP |

* This number is the result of the conversion of KH = 8000.

## 8.5 Sample Programs

Sections 8.5.1 through 8.5.3 provide a few sample programs that you can enter and test in all three methods of representation on a programmer.

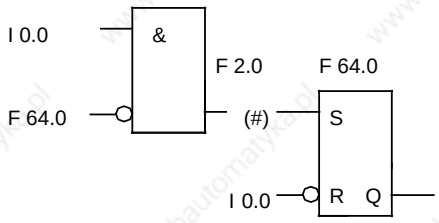
### 8.5.1 Momentary-Contact Relay/Edge Evaluation

| Example | Circuit Diagram |
|---|---|
| On each leading edge of the signal at input I 0.0, the AND condition “A I 0.0 and AN F 64.0” is satisfied; the RLO is “1”. This sets flags F 64.0 and F 2.0 (“edge flags”).<br>In the next processing cycle, the AND condition “A I 0.0 and AN F 64.0” is not satisfied since flag F 64.0 has already been set.<br>Flag 2.0 is reset.<br>Therefore, flag F 2.0 is “1” for only one program run.<br>When input I 0.0 is switched off, flag F 64.0 is reset.<br>This resetting prepares the way for evaluation of the next leading edge of the signal at input I 0.0. |  |

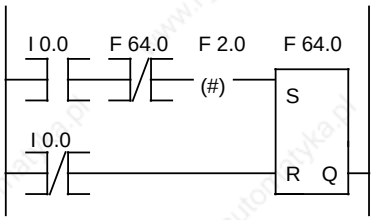
**STL**

```
A     I    0.0
AN    F   64.0
=     F    2.0
S     F   64.0
AN    I    0.0
R     F   64.0
NOP   0
```

**CSF**




**LAD**

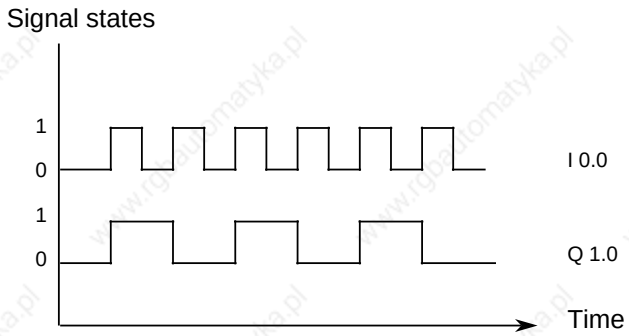



### 8.5.2 Binary Scaler/Binary Divider

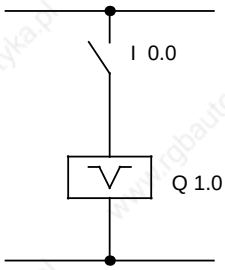
This section describes how to program a binary scaler.

**Example:** The binary scaler (output Q 1.0) changes its state each time I 0.0 changes its signal state from “0” to “1” (leading edge). Therefore, half the input frequency appears at the output of the flip-flop.

**Timing Diagram**



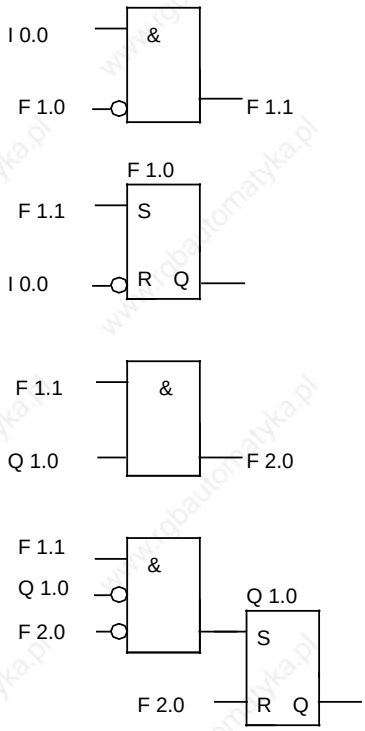

**Circuit Diagram**



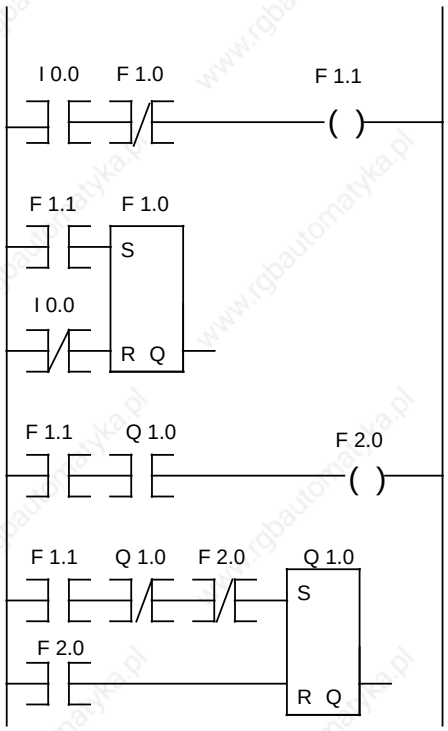

**STL**

```
A     I     0.0
AN    F     1.0
=     F     1.1
***
A     F     1.1
S     F     1.0
AN    I     0.0
R     F     1.0
NOP 0
***
A     F     1.1
A     Q     1.0
=     F     2.0
***
A     F     1.1
AN    Q     1.0
AN    F     2.0
S     Q     1.0
A     F     2.0
R     Q     1.0
NOP 0
***
```

**CSF**




**LAD**




## Note

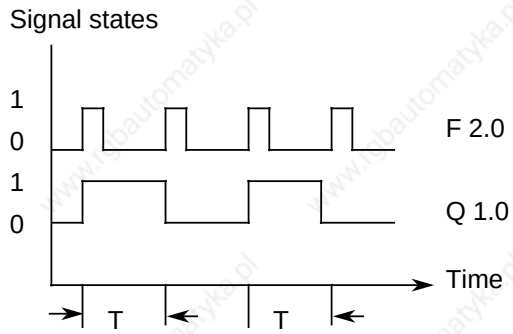
Output in CSF or LAD is possible only if you enter the segment boundaries "***" when programming in STL.

## 8.5.3 Clock/Clock-Pulse Generator

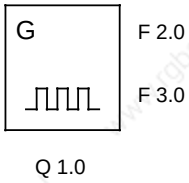
This subsection describes how to program a clock-pulse generator.

**Example:** A clock-pulse generator can be implemented using a self-clocking timer that is followed in the circuit by a binary scaler. Flag F 2.0 restarts timer T 7 each time it runs down, i.e., flag F 2.0 is “1” for one cycle each time the timer runs down. The pulses of flag F 2.0 applied to the binary scaler result in a pulse train with pulse duty factor 1:1 at output Q 1.0. The period of this pulse train is twice as long as the time value of the self-clocking timer.

**Timing Diagram**



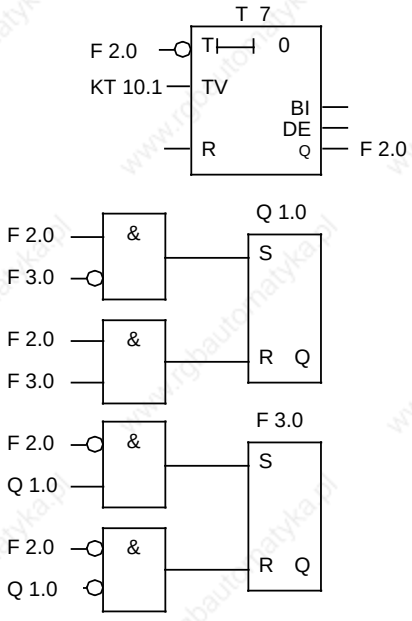

**Circuit Diagram**



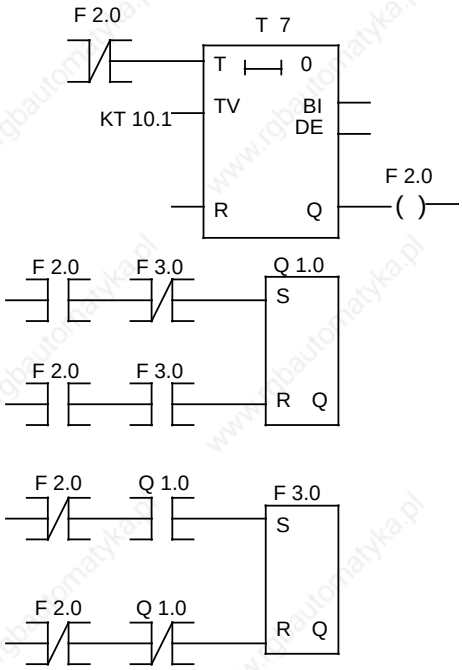

**STL**

```
AN    F     2.0
L     KT  010.1
SD    T     7
NOP   0
NOP   0
NOP   0
A     T     7
=     F     2.0
***
A     F     2.0
AN    F     3.0
S     Q     1.0
A     F     2.0
A     F     3.0
R     Q     1.0
NOP   0
***
AN    F     2.0
A     Q     1.0
S     F     3.0
AN    F     2.0
AN    Q     1.0
R     F     3.0
NOP   0
```

**CSF**




**LAD**



## 9 Integrated Blocks and Their Functions


| | | |
|---|---|---|
| 9.1 | Assigning Internal Functions to DB1, for CPU 103 Version 8MA03 and Higher | 9 - 1 |
| 9.1.1 | Configuration and Default Settings for DB1 | 9 - 1 |
| 9.1.2 | Setting the Address for the Parameter Error Code in DB1 | 9 - 2 |
| 9.1.3 | Assigning Parameters in DB1 | 9 - 4 |
| 9.1.4 | Rules for Setting Parameters in DB1 | 9 - 4 |
| 9.1.5 | How to Recognize and Correct Parameter Errors | 9 - 6 |
| 9.1.6 | Transferring DB1 Parameters to the Programmable Controller | 9 - 9 |
| 9.1.7 | Reference Guide for Setting Parameters in DB1 | 9 - 10 |
| 9.1.8 | Defining System Characteristics in DB1 | 9 - 11 |
| 9.2 | Integrated Function Blocks, for CPU 102 Version 8MA02 and Higher | 9 - 11 |
| 9.2.1 | Code Converter : B4 - FB240 - | 9 - 12 |
| 9.2.2 | Code Converter : 16 - FB241 - | 9 - 12 |
| 9.2.3 | Multiplier : 16 - FB242 - | 9 - 13 |
| 9.2.4 | Divider : 16 - FB243 - | 9 - 13 |
| 9.2.5 | Analog Value Conditioning Modules FB250 and FB251 | 9 - 14 |
| 9.3 | Integrated Organization Blocks | 9 - 14 |
| 9.3.1 | Scan Time Triggering OB31, for CPU 103 and Higher | 9 - 14 |
| 9.3.2 | Battery Failure OB34 | 9 - 14 |
| 9.3.3 | OB251 PID Algorithm, for CPU 103 Version 8MA02 and Higher | 9 - 15 |

**Figures**


9-1 DB1 with Default Parameters . . . . . 9 - 1
9-2 Inputting the Address for the Parameter Error Code . . . . . 9 - 3
9-3 Parameter Error Codes and Their Meaning . . . . . 9 - 7
9-4 Erroneous Parameter Assignment in DB1 . . . . . 9 - 8
9-5 Inputting the System Data Parameter . . . . . 9 - 11
9-6 Calling Up the OB251 PID Algorithm . . . . . 9 - 15
9-7 Block Diagram of the PID Controller . . . . . 9 - 16
9-8 Principle of Interval Sampling . . . . . 9 - 21
9-9 Process Schematic . . . . . 9 - 22


**Tables**


9-1 Parameter Blocks and Their IDs . . . . . 9 - 2
9-2 Call and Parameter Assignments of FB240 . . . . . 9 - 12
9-3 Call and Parameter Assignments of FB241 . . . . . 9 - 12
9-4 Call and Parameter Assignments of FB242 . . . . . 9 - 13
9-5 Call and Parameter Assignments of FB243 . . . . . 9 - 13
9-6 Legend for the Block Diagram of the PID Controller . . . . . 9 - 16
9-7 Description of the Control Bits in Control Word ”STEU” . . . . . 9 - 17
9-8 Structure of the Controller DB . . . . . 9 - 19

# 9 Integrated Blocks and Their Functions

## 9.1 Assigning Internal Functions to DB1, for CPU 103 Version 8MA03 and Higher

You can program the following CPU functions:

- Using the integral real-time clock (see chapter 12)
- Exchanging data via SINEC L1 (see chapter 13)
- Changing polling interval for time-controlled program processing (OB 13) (see chapter 7)
- Assigning system parameters (see chapter 9)
- Setting the address for the parameter error code (see chapter 9)

To assign parameters to these functions, you must configure data block 1 (DB1).

### 9.1.1 Configuration and Default Settings for DB1

To make it easier for you to assign parameters, data block 1 is already integrated in the CPU with default parameters. After performing an overall reset, you can load the default DB1 from the programmable controller into your programmer and display it on the screen (see Figure 9-1). The character string “DB1” must remain before the parameter blocks and be followed by at least one filler (such as a blank space or a comma).

```
  0:     KS  = 'DB1 SL1: SLN 1       SF ';
 12:     KS  = 'DB2  DW0   EF DB3  DW0  ';
 24:     KS  = ' KBE MB100 KBS MB101    ';
 36:     KS  = 'PGN 1  ; #CLP: CF 0     ';
 48:     KS  = 'CLK DB5  DW0   STW      ';
 60:     KS  = 'MW102       STP Y SAV Y ';
 72:     KS  = 'OHE N    SET 4 01.04.92 ';
 84:     KS  = '12:10:00     TIS 4      ';
 96:     KS  = '01.04. 13:00:00     OHS ';
108:     KS  = '000000:00:00 # ; SDP: WD';
120:     KS  = ' 500 ; TFB: OB13  100  ';
132:     KS  = ' ;  END                 ';
```

**Figure 9-1. DB1 with Default Parameters**

This preset DB1 has one parameter block for each function. Each parameter block begins with a block ID (shown in Figure 9-1 in the shaded background). The block ID is followed by a colon. The individual parameters for each function are contained in these parameter blocks.

Each parameter block begins with a block ID followed by a colon. This colon must be followed by at least one filler (such as a blank space or a comma). A semicolon must be at the end of each parameter block with at least one filler between the semicolon and the next block ID.

The parameter blocks listed in Table 9-1 are used for the S5-100U.

**Table 9-1. Parameter Blocks and Their IDs**

| Block ID | Explanation/Default Setting |
|---|---|
| `'DB1 ';` | Start ID |
| `'SL1: ';` | ***S***INEC ***L1***: Parameter block for SINEC L1 configuration / (see chapter 13) |
| `'CLP: ':` | ***Cl***ock-***P***arameters: Parameter block for integral time clock/ clock function **not** activated (see chapter 12) |
| `'SDP: ';` | **S**ystem-***D***ependent ***P***arameter: Parameter block for system specifications / default setting for cycle time monitoring is 500 ms (see section 9.1.8) |
| `'TFB: ';` | ***T***imer ***F***unction ***B***locks: Parameter block for time-controlled program processing: OB13 is called up every 100 ms. (see chapter 7) |
| `'ERT: ';` | ***E***rror ***R***e***T***urn: Address for parameter error code / no default setting (see section 9.1.2) |
| `'END ';` | ***END*** block ID for DB1 |

The sequence of the parameters in DB1 is not fixed. A semicolon must be at the end of each parameter block with at least one filler between the semicolon and the next block ID.

The structure of the following parameter blocks is described here in detail.

- ERT: (Error code position)
- SDP: (System specifications)

The parameter blocks that are not discussed here are explained in the chapters that describe their functions.

## 9.1.2 Setting the Address for the Parameter Error Code in DB1

For the following reasons, we recommend that you use this example when you start setting your parameters:

- Parameter block “ERT:” is the only block with no default parameters in DB1. You must therefore enter all the parameters. We will explain the rules for assigning parameters step by step, so that you can learn the rules quickly.
- The correctly input “ERT:” parameter block makes it easy for you to correct parameter setting errors; therefore, you should complete this block in DB1 before you change or add other parameters.

  The error parameter block is only important during the start-up phase. You should erase it during “normal” operation because it takes up a lot of memory space.

To help find parameter errors more easily and to help correct them, you can ask the programmable controller to output error messages in a coded form. All you have to do is to tell the programmable controller where it should store the error code. Make this input in parameter block “ERT:” of DB1.

The error code can be stored in either of the following locations:

- In flagwords
- In data words in a data block

**How to Proceed:**

1. Perform an overall reset on the programmable controller.
2. Display DB1 on the programmer.
3. Position the cursor on the E of the “END” ID at the end of the default DB1.
4. Enter the characters that are highlighted in Figure 9-2.

| DB1 | Explanation |
|---|---|
| 0: KS = 'DB1 SL1: SLN 1 SF ';<br>12: KS = 'DB2 DW0 EF DB3 DW0 ';<br>24: KS = ' KBE MB100 KBS MB101 ';<br>36: KS = 'PGN 1 ; #CLP: CF 0 ';<br>48: KS = 'CLK DB5 DW0 STW ';<br>60: KS = 'MW102 STP Y SAV Y ';<br>72: KS = 'OHE N SET 4 01.04.92 ';<br>84: KS = '12:10:00 TIS 4 ';<br>96: KS = '01.04. 13:00:00 OHS ';<br>108: KS = '000000:00:00 # ; SDP: WD';<br>120: KS = ' 500 ; TFB: OB13 100 ';<br>132: KS = ' ; ERT: ERR MW1 ; END ']; | The parameter error code is stored in flag word MW1 after start-up. |

**Figure 9-2. Inputting the Address for the Parameter Error Code**

5. Use the following check list to make sure your entries are correct.
   - Is the block ID “ERT:” terminated by a colon? . . . . . .
   - Is at least 1 filler (a blank space in Figure 9-2) added after the colon? . . . . . .
   - Is the parameter name (ERR) entered correctly? . . . . . .
   - Does at least 1 filler (a blank space) follow the parameter name? . . . . . .
   - Is the argument (MW1) entered correctly? . . . . . .
   - Does at least 1 filler (a blank space) follow the argument? . . . . . .
   - Does a semicolon (;) indicate the block end? . . . . . .
   - Does DB1 end with the end ID “END” followed by a space? . . . . . .
6. Transfer the changed DB1 to the programmable controller.
7. Switch the programmable controller from STOP to RUN.
   - The programmable controller accepts the changed DB1.

If you did not store the parameter block “ERT:” in DB1, you can localize the error in the ISTACK if there was an incorrect parameter setting. However, you will not know what type of error is present. The same applies if you made an error when you input the parameter block “ERT:”

### 9.1.3 Assigning Parameters in DB1

As discussed in section 9.1.2, you use the following steps to change or expand the preset values of DB1:

1. Display the default DB1, with its parameter block “ERT:” on the programmer.
2. Position the cursor on the desired parameter block.
3. Change or expand the parameters.

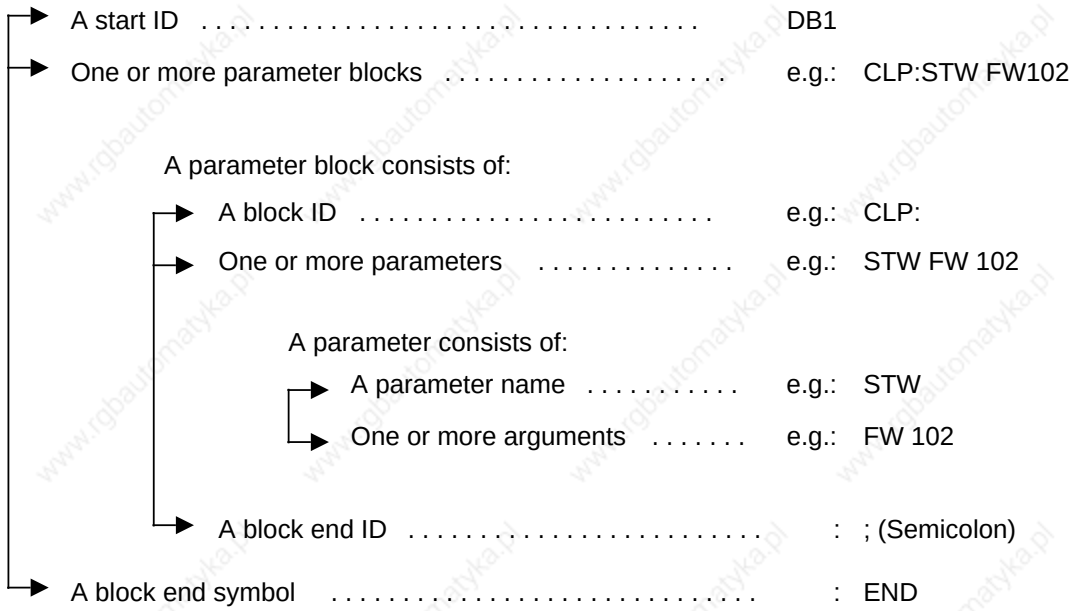
   (for an explanation and possible parameter values see section 9.1.7)
4. Transfer the changed DB1 to the programmable controller.
5. Switch the programmable controller from STOP to RUN.

Changed DB1 parameters are accepted.

> **Note**
>
> If the CPU recognizes an error in DB1, then it remains in the STOP mode (red LED lights up) even after a switch from STOP to RUN.

### 9.1.4 Rules for Setting Parameters in DB1

DB1 consists of the following:



In the following section are the rules for changing or expanding entire parameter blocks. Follow these steps or the CPU will not understand what you have entered.

1. Enter the start ID “DB1”, followed by a filler.
   - DB1 must begin with the start ID “DB1”. Do not separate the three characters from each other. After the start ID, there must be at least one filler. Use a blank space or a comma as a filler.

2. Enter the block ID for the parameter block, followed by a filler.
   - The start ID and filler are followed by the block ID for the parameter block. The sequence of the parameter blocks in DB1 is random. The block ID identifies a block and its corresponding parameter. The block ID “SL1”, for example, stands for the SINEC-L1 parameter. You must enter a colon immediately after the block ID. If the colon is missing, then the CPU skips this block and displays an error message. You must add at least one filler after the colon of a block ID.

3. Enter the parameter name, followed by a filler.
   - The parameter name comes next. Parameter names are names for single parameters within a parameter block. Within a block, the first four characters of a parameter name must be different from each other. After the parameter name, you must add at least one filler.

4. Enter the argument that is attached to the parameter name, followed by a filler.
   - At least one argument is attached to each parameter name. An argument is either a number or a STEP 5 operand that you must enter. If several arguments belong to a parameter name, then every argument must be followed by at least one filler (even the last one).

5. Enter a semicolon (; ) to identify the block end, followed by a filler.
   - After the semicolon, you must enter at least one filler. Leaving out the semicolon leads to misinterpretation in the CPU.

6. Enter additional parameter blocks after the semicolon.
   - (Use steps 2 through 5 to create additional parameter blocks.)

7. Enter the end ID “END”.
   - This identifies the end of DB1. If you forget to enter an end ID, this leads to errors in the CPU.

The preceding steps present the minimal requirements for setting the parameters. Beyond that, there are additional rules that make it easier for you to assign parameters.

For example:

- You have the ability to add comments.
- You can expand the German mnemonics used as parameter names by using plain English text.

Comments can be added anywhere a filler is allowed. The comment symbol is the pound (#) sign. The comment symbol must be placed at the beginning and at the end of your comment. The text between two comment symbols may not contain an additional #.

Example: #Comment# .

At least one filler must follow the comment.
If you wish to change the default settings in parameter blocks SL1: or CLP:, you must first of all overwrite the two comment characters (#) with blanks. If you fail to do this, the changes are ignored.
If you wish to retain the default settings for one of the two parameter blocks, you must place it between comment characters (overwrite blanks with "#").

In order to make it easier to read parameter names, you can add as many characters as you wish if you add an underscore (_) after the abbreviated parameter name.
Example: SF becomes SF_SENDMAILBOX.
At the end of the input, you must add at least one filler.

There is a rule of thumb that will help you check DB1. You should include at least one filler in the following instances:

- After the start ID
- Before and after the block ID, parameter name, argument, and semicolon

## 9.1.5 How to Recognize and Correct Parameter Errors

If an error occurs while assigning parameters and the programmable controller does not go to the "RUN" mode, you have two possibilities for recognizing errors:

- By using a parameter error code
- By using the analysis function "ISTACK"

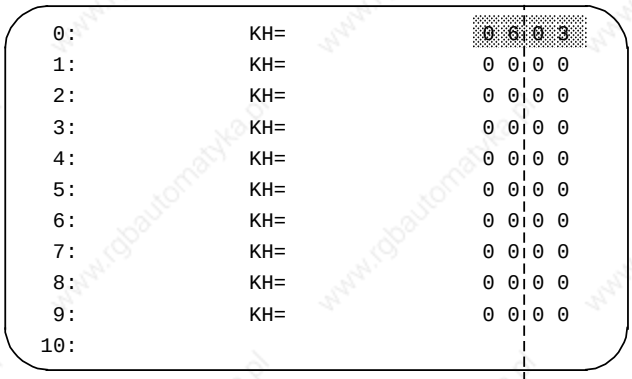
Both possibilities are described below.

**Scanning the Parameter Error Code**

If you have entered a start address for the parameter error code in parameter block "ERT:" of DB1 (see section 9.1.2), then you can retrieve the cause of the error, and the error location information at this address.

The entire error code occupies 10 data words or 20 flag bytes. In the following examples and tables, we assume that the error code is stored in a data block starting with data word 0. The error code occupies DW0 through DW9. In the "Flag" operand area, this corresponds to FW0 through FW19.

Example: You entered the start address DB3 DW0 in parameter block “ERT:”. The parameters set in DB1 have already been transferred to the programmable controller. Then you continue to set parameters in DB1. While attempting to transfer the changed DB1 parameters to the programmable controller, you find out that the programmable controller remains in the STOP mode. You suspect that the reason for this is a parameter error. To find the error, display DB3 on the programmer. The entire contents of DB3 appear on the screen. DW0 through DW9 contain the code for the parameter error. In the following figure, you see how your screen could look. Below the screen display is a complete list of parameter error codes and their meanings.




Screen display with parameter error codes

| Cause of the error (which error occurred?) | DL | DR | Location of error (in which parameter block did the error occur?) |
|---|---|---|---|
| No error | 00 | 00 | Error cannot be assigned to any block |
| Start or end ID is missing | 01 | 01 | |
| Comment not closed off correctly | 02 | 02 | |
| Before END; semicolon missing in front of END | | 03 | SL1: SINEC L1 |
| Syntax error - block ID | 03 | | |
| Syntax error - parameter | 04 | 06 | CLP: Clock parameter |
| Syntax error - argument | 05 | | |
| Range exceeded | 06 | | |
| in an argument | | 09 | TFB: Timer function block |
| Parameter combination is not allowed | 07 | | |
| | | 11 | SDP: System data parameter |
| Not defined | 08 | | |
| Not defined | 09 | | |
| DB is not present | 10 | | |
| Not enough space in DB | 11 | 99 | ERT: Error return |
| Error inputting weekday | 12 | F0 | Error can not be assigned to any block |
| Error in the date | 13 | . | |
| Error in inputting time | 14 | . | |
| Irregular time format in parameter blocks (24h/12h mode) | 15 | . FF | |

**Figure 9-3. Parameter Error Codes and Their Meaning**

**Locating Parameter Errors in "ISTACK"**

If the CPU recognizes an error in DB1 in the initial start-up, then the CPU remains in the STOP mode and stores a message in "ISTACK" describing where the error happened. The "ISTACK" contains the absolute error address as well as the relative error address.

The STEP Address Counter (SAC) in the ISTACK points either **to** the address that contains the incorrect input or **in front of** the address that contains the incorrect input. These are byte addresses.

**Example:** Your inputs into DB1 are as follows. The position shaded contains an error.

```
  0:     KS  = 'DB1 SL1: SLN 40     SF  ';
 12:     KS  = 'DB2  DW0   EF DB3  DW0  ';
 24:     KS  = ' KBE MB100 KBS MB101    ';
 36:     KS  = 'PGN 1  ; #CLP: CF 0     ';
 48:     KS  = 'CLK DB5  DW0   STW      ';
 60:     KS  = 'MW102       STP Y SAV Y ';
 72:     KS  = 'OHE N    SET 4 01.04.92 ';
 84:     KS  = '12:10:00     TIS 4      ';
 96:     KS  = '01.04. 13:00:00     OHS ';
108:     KS  = '000000:00:00 # ; SDP: WD';
120:     KS  = ' 500 ;  TFB: OB13  100  ';
132:     KS  = ' ; END                  ';
```

The decimal numbers in front of each input line represent the word address for the first character that can be entered for that respective line. Each word consists of two characters (2 bytes).

**Figure 9-4. Erroneous Parameter Assignment in DB1**

The error causes ISTACK to display the following addresses.

- The absolute (error) address: $82F2_H$ (absolute SAC)
- The relative (error) address: $000C_H$ (relative SAC)

So that you can locate the error in DB1 exactly, you must convert the relative byte address that is displayed in hexadecimal format into a decimal word address. Decimal format is required because the programmer displays a DB in words.

| | | | | | | |
|---|---|---|---|---|---|---|
| $000C_H$ | = | $12_D$ | $12_D$ | : | $2_D$ = | $6_D$ |
| Hexadecimal byte address | | Decimal byte address | | | | Decimal word address |

The information displayed in the chart above shows that the error occurred after address 0 and before address 12. In Figure 9-4, argument 40 occupies address 6; the "40" is an incorrect entry. The error is due to a range violation.

### 9.1.6 Transferring DB1 Parameters to the Programmable Controller

Unlike other data blocks, DB1 is processed only one time. This occurs when a cold restart is performed on the programmable controller. This was done so that DB1 could handle certain special functions.

One such special function is the assignment of parameters in the programmable controller with the help of DB1. Setting parameters means that you enter parameters in DB1 for those internal functions that your programmable controller should work with.

The programmable controller's operating system accepts these inputs into DB1 only when there is a cold restart. You must perform a cold restart anytime you make changes to DB1. You can perform a cold restart by switching from Power OFF to Power ON or from STOP to RUN.

The programmable controller accepts the parameters from DB1 and stores them in the system data area.

**Note**

The CPU remains in the STOP mode if a parameter assignment error is found during start-up. The red LED lights up on the operator panel and ISTACK displays a DB1 addressing error.

### 9.1.7 Reference Guide for Setting Parameters in DB1

| Parameter | Argument | Meaning |
|---|---|---|
| **Block ID: SL1:** | | **SINEC L1 (SL1)** |
| SLN | p | Slave number |
| SF | DBx DWy | Location of Send Mailbox |
| EF | DBxDWy | Location of Receive Mailbox |
| KBE | MBy | Location of Coordination Byte “Receive” |
| KBS | MBy | Location of Coordination Byte “Send” |
| PGN | p | Programmer bus number |
| p=1 to 3 | x=2 to 255 | y=0 to 255 |
| **Block ID:SDP:** | | **System-Dependent-Parameter (SDP)** |
| WD | p | Number of timers being processed(***W***atch-***D***og-***T***imer) |
| p=1 to 2550 | | |
| **Block ID: TFB:** | | **Timer-Funktions-Baustein (TFB)** |
| OB13 | p | Intervals (ms) at which OB13 is called up and is processed |
| p=0 to 655350 (State in 10-ms steps) | | |
| **Block ID: CLP:** | | **ClockParameters (CP)** |
| CF | p | Inputting the correction factor (***C***orrection ***F***actor) |
| CLK | DBxDWy,MWz,EWv or AWv | Location of the clock data (***CL***oc***K*** Data) |
| STW | DBxDWy,MWz,EWv or AWv | Location of the status word (***ST***atus ***W***ord) |
| STP | J/Y/N | Updating the clock during STOP (***ST***o***P)*** |
| SAV | J/Y/N | Saving the clock time after the last change from RUN to STOP or Power OFF (***SAV***e) |
| OHE | J/Y/N | Enabling the operating hours counter (***O***peration ***H***our counter ***E***nable) |
| SET | wd dd.mm.jj hh:mn:ss[1] AM/PM[2] | Setting the clock time and date |
| TIS | wd dd.mm. hh:mn:ss[1] AM/PM[2] | Setting the prompting time (***TI***mer ***S***et) |
| OHS | hhhhhh:mn:ss[1] | Setting the operating hours counter (***O***peration ***H***our counter ***S***et) |

| | | | |
|---|---|---|---|
| wd | =1 to 7 (weekday = Sun..Sat) | | p=– 400 to 400 |
| dd | =01 to 31 (day) | | v=0 to 126 |
| mm | =01 to 12 (month) | x=2 to 255 | |
| yy | =0 to 99 (year) | | y=0 to 255 |
| hh | =00 to 23 (hours) | | z=0 to 254 |
| mn | =00 to 59 (minutes) | | j/J=ja(yes) |
| ss | =00 to 59 (seconds) | | y/Y=yes |
| hhhhhh | =0 to 999999 (hours) | | n/N=no |

[1] If an argument such as seconds, for example, is not to be entered, input XX. The clock continues to run with the updated data. The TIS parameter block does not acknowledge this argument..

[2] If you input AM or PM after the clock time, the clock runs in the 12-hour mode. If you omit this argument, the clock runs in the 24-hour mode. You must use the same time mode in the SET and TIS parameter blocks.

### 9.1.8 Defining System Characteristics in DB1

Each cyclical program processing triggers the beginning of a monitoring period. If the cycle trigger is not retriggered during the monitoring period, the programmable controller is forced into the STOP mode and disables the output modules. The default for the monitoring time is set to 500 ms in DB1. You can increase the cycle time monitoring in the parameter block SDP.

**Example:** You wish to increase the monitoring time to 700 ms since your user program is very large.

**How to Proceed:**

1. Display DB1 on the programmer.
2. Change the parameter block “SDP” as shown in Figure 9.5.
   - Position the cursor on the arguments for the parameter
   - Overwrite the arguments
3. Transfer the changed DB1 to the programmable controller.
4. Switch the programmable controller from STOP to RUN. The programmable controller now accepts the changed parameters.

```
  0:     KS  = 'DB1 SL1: SLN 1       SF ';
 12:     KS  = 'DB2  DW0   EF DB3  DW0  ';
 24:     KS  = ' KBE MB100 KBS MB101    ';
 36:     KS  = 'PGN 1  ; #CLP: CF 0     ';
 48:     KS  = 'CLK DB5  DW0   STW      ';
 60:     KS  = 'MW102       STP Y SAV Y ';
 72:     KS  = 'OHE N    SET 4 01.04.92 ';
 84:     KS  = '12:10:00     TIS 4      ';
 96:     KS  = '01.04. 13:00:00     OHS ';
108:     KS  = '000000:00:00 # ; SDP: WD';
120:     KS  = ' 700 ;  TFB: OB13  100  ';
132:     KS  = ' ; END                  ';
```

**Figure 9.5. Inputting the System Data Parameter**

You can also set the cycle monitoring time in OB31 (see section 9.3.1).

## 9.2 Integrated Function Blocks, for CPU 102 Version 8MA02 and Higher

Some standard function blocks are integrated in your S5-100U. You can call up these blocks in your control program with the commands “JU FB” or “JC FB x”. The character “x” stands for the block number.

**Overview**:

| Block No. | **FB240** | **FB241** | **FB242** | **FB243** | **FB250** | **FB251** |
|---|---|---|---|---|---|---|
| Block name | COD:B4 | COD:16 | MUL:16 | DIV:16 | RLG:AI | RLG:AQ |
| Call length (in words) | 5 | 6 | 7 | 10 | 10 | 9 |
| Processing time (in ms) | < 0.6 | < 1.0 | < 0.9 | < 2.1 | 2.4 | 4.8 |

### 9.2.1 Code Converter : B4 - FB240 -

Use function block FB240 to convert a number in BCD (4 tetrads) with sign to a fixed-point binary number (16 bits).

You must change a two-tetrad number to a four-tetrad number before you convert it.

- If a tetrad is not in the BCD defined range, then FB240 displays the value “0”. An error bit message does not follow.

**Table 9-2. Call and Parameter Assignments of FB240**

| Parameter | Meaning | Type | Assignment | STL |
|---|---|---|---|---|
| BCD | BCD number | I W | 0 to 9999 | : JU FB240<br>NAME : COD:B4<br>BCD :<br>SBCD :<br>DUAL : |
| SBCD | Sign of the BCD number | I BI | “1” for “-”<br>“0” for “+” | |
| DUAL | Fixed-point number (KF) | Q W | 16 bits “0” or “1” | |

### 9.2.2 Code Converter : 16 - FB241-

Use function block FB 241 to convert a fixed-point binary number (16 bits) to a number in BCD code with additional consideration of the sign. An eight-bit binary number must be transferred to a 16-bit word before conversion.

**Table 9-3. Call and Parameter Assignments of FB241**

| Parameter | Meaning | Type | Assignment | STL |
|---|---|---|---|---|
| DUAL | Binary number | I W | -32768 to+32767 | : JU FB241<br>NAME : COD:16<br>DUAL :<br>SBCD :<br>BCD2 :<br>BCD1 : |
| SBCD | Sign of the BCD number | I BI | “1” for “-”<br>“0” for “+” | |
| BCD2 | BCD number 4th and 5th tetrads | Q BY | 2 tetrads | |
| BCD1 | BCD number tetrads 0 to 3 | Q W | 4 tetrads | |

### 9.2.3 Multiplier : 16 - FB242 -

Use function block FB 242 to multiply one fixed-point binary number (16 bits) by another. The product is represented by two fixed-point binary numbers (16 bits each). The result is also scanned for zero. An eight-bit number must be transferred to a 16-bit word prior to multiplication.

**Table 9-4. Call and Parameter Assignments of FB242**

| Parameter | Meaning | Type | Assignment | STL |
|---|---|---|---|---|
| Z1 | Multiplier | I W | -32768 to+32767 | : JU FB242<br>NAME : MUL:16<br>Z1 :<br>Z2 :<br>Z3=0 :<br>Z32 :<br>Z31 : |
| Z2 | Multiplicand | I W | -32768 to+32767 | |
| Z3=0 | Scan for zero | Q BI | “0” : product is zero | |
| Z32 | Product high-word | Q W | 16 Bits | |
| Z31 | Product low-word | Q W | 16 Bits | |

### 9.2.4 Divider : 16 - FB243 -

Use function block FB 243 to divide one fixed-point binary number (16 bits) by another. The result (quotient and remainder) is represented by two fixed-point binary numbers (16 bits each).

The divisor and the result are also scanned for zero. An eight-bit number must be transferred to a 16-bit word prior to division.

**Table 9-5. Call and Parameter Assignments of FB243**

| Parameter | Explanation | Type | Assignment | STL |
|---|---|---|---|---|
| Z1 | Dividend | I W | -32768 to+32767 | : JU FB243<br>NAME : DIV:16<br>Z1 :<br>Z2 :<br>OV :<br>FEH :<br>Z3=0 :<br>Z4=0 :<br>Z3 :<br>Z4 : |
| Z2 | Divisor | I W | -32768 to+32767 | |
| OV | Overflow bit | Q BI | “1” : overflow | |
| FEH | | Q BI | “1” : division by zero | |
| Z3=0 | Scan for zero | Q BI | “0”: quotient is zero | |
| Z4=0 | Scan for zero | Q BI | “0”: remainder is zero | |
| Z3 | Quotient | Q W | 16 bits | |
| Z4 | Remainder | Q W | 16 bits | |

### 9.2.5 Analog Value Conditioning Modules FB250 and FB251

Function block FB250 reads in an analog value from an analog input module and outputs a value XA in the scale range specified by the user.

Function block FB251 allows you to output analog values to analog output modules. Values from the range between the “UGR” (lower limit) parameters and the “OGR” (upper limit) parameters are converted to the nominal range of the selected module.

You will find more information on the following topics in section 11.6:

- Calling up and setting parameters in FB250.
- Calling up and setting parameters in FB251.
- An example of analog value processing with FB250 and FB251.

## 9.3 Integrated Organization Blocks

### 9.3.1 Scan Time Triggering OB31, for CPU 103 and Higher

A scan time monitor monitors the program scan time. If program scanning takes longer than the specified scan monitoring time, the CPU goes into the STOP mode. This can happen when one of the following errors occurs:

- The control program is too long.
- The program enters a continuous loop.

You can retrigger the scan time monitor at any point in the control program by calling up OB31. Calling up this block restarts the scan time monitor.

**Call up OB31**

- Prerequisite: SYSTEM COMMANDS “YES” has been specified on the programmer.
- JU OB31 can be programmed at any point in the control program.

**Programming**

One statement in OB31 is sufficient, e.g. “BE” to make the retriggering effective. Other statements are also possible.

### 9.3.2 Battery Failure OB34

The CPU constantly checks the status of the battery in the power supply. If a battery fails (BAU), OB34 is processed before every cycle until the battery is replaced. You can program the reaction of the programmable controller to battery failure in OB34. If OB34 is not programmed, there is no reaction.

### 9.3.3 OB251 PID Algorithm, for CPU 103 Version 8MA02 and Higher

A PID algorithm is integrated in the operating system of the S5-100U. OB251 helps you use this algorithm to meet your needs.

Before calling up OB251, you must first open a data block called the controller DB. It contains the controller parameters and other controller specific data. The PID algorithm must be called up periodically to generate the manipulated variable. The more closely the scan time is maintained, the more accurately the controller fulfills its task. The control parameters specified in the controller DB must be adapted to the scan time.

You should always call OB251 from the time OB (OB13). You can set time OBs at a call up interval ranging between 10 ms and 655,350 ms. The PID algorithm requires no more than 1.7 ms to process.

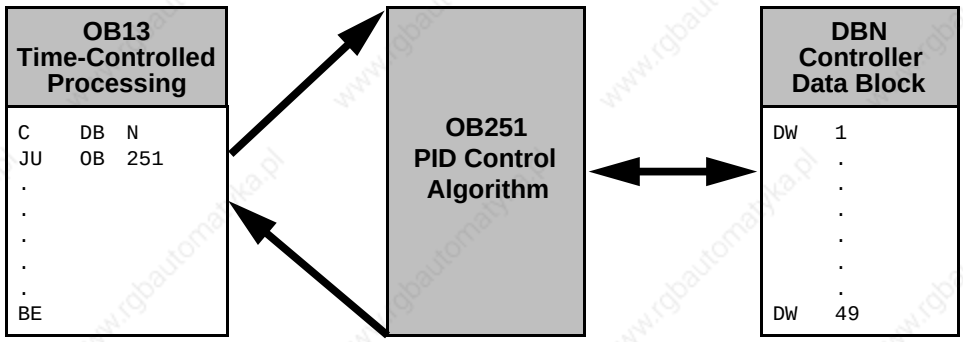



**Figure 9-6. Calling Up the OB251 PID Algorithm**

The continuous action controller is designed for controlled systems such as those present in process engineering for controlling pressure, temperature, or flow rate.

The “R” variable sets the proportional element of the PID controller. If proportional action is required, most controller designs use the value R = 1.

The individual Proportional action, Integral action, and Derivative action elements can be deactivated via their parameters (R, TI, and TD) by presetting the pertinent data words to zero. This enables you to implement all required controller structures without difficulty, e.g., PI, PD, or PID controllers.

You can forward the system deviation XW or, using the XZ input, any disturbance variable or the inverted actual value X to the derivative action element. Specify a negative K value for a reverse acting controller.

When the manipulated information (dY or Y) is at a limit, the integral action component is automatically deactivated in order not to impair the dynamic response of the controller.

The switch settings in the block diagram are implemented by setting the respective bits in control word “STEU”.

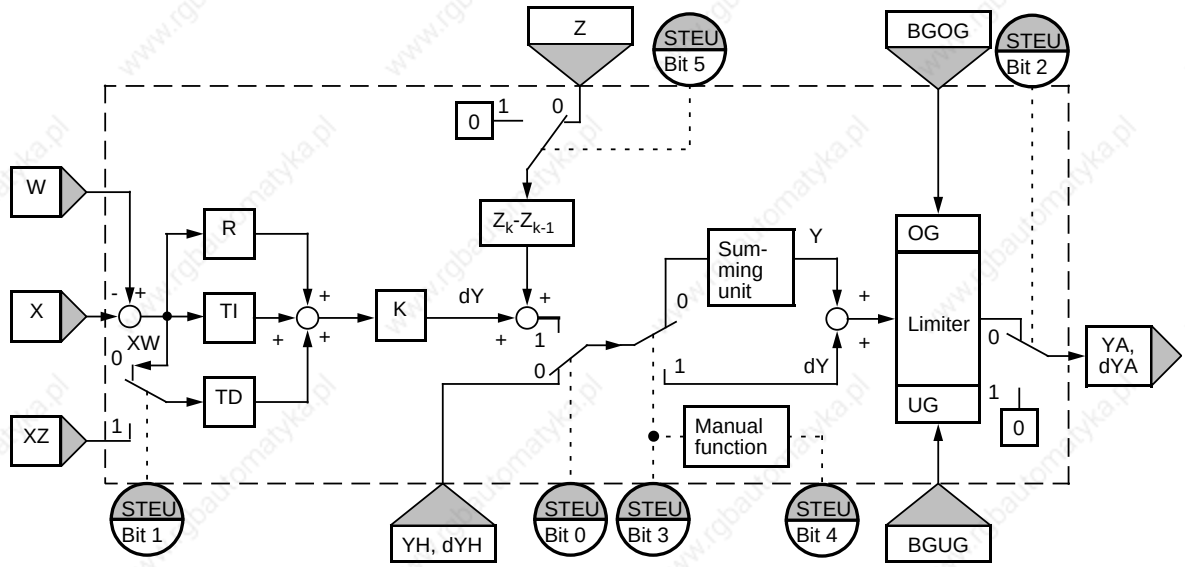



**Figure 9-7. Block Diagram of the PID Controller**

**Table 9-6. Legend for the Block Diagram of the PID Controller**

| Designation | Explanation |
|---|---|
| K | Proportional coefficient: K>0 direct acting<br>K<0 reverse acting |
| R | R parameter (usually 1000) |
| TA | Scan time |
| TN | Integral-action time |
| TV | Derivative-action time |
| TI | Constant TI TI=Scan time TA/Integral action time TN |
| TD | Constant TD TD=Derivative action time TV/Scan time TA |
| W | Setpoint |
| STEU | Control word |
| YH, dYH | Output value: YH Control Word Bit 3=0<br>dYH Control Word Bit 3=1 |
| Z | Disturbance variable |
| XW | System deviation |
| X | Actual value |
| XZ | Substitute value for system deviation |
| Y, dY | Manipulated variable, manipulated increments |
| BGOG | Upper limit of the manipulated variable |
| BGUG | Lower limit of the manipulated variable |
| YA, dYA | Output word : YA Control Word Bit 3=1<br>dYA Control Word Bit 3=0 |

**Table 9-7. Description of the Control Bits in Control Word "STEU"**

| Control Bit | Name | Signal State | Description |
|---|---|---|---|
| 0 | AUTO | 0 | Manual mode<br>The following variables are updated in Manual mode:<br>1) $X_K$, $XW_{K-1}$ and $PW_{K-1}$<br>2) $XZ_K$, $XZ_{K-1}$ and $PZ_{K-1}$, when STEU bit 1=1<br>3) $Z_K$ and $Z_{K-1}$, when STEU bit 5=0<br>Variable $dD_{K-1}$ is set to 0: The algorithm is not computed. |
| | | 1 | Automatic mode |
| 1 | XZ EIN | 0<br>1 | $XW_k$ is forwarded to the derivative action element. The XZ input is ignored. A variable other than $XW_k$ is forwarded to the derivative action element . |
| 2 | REG AUS | 0<br>1 | Normal controller processing<br>When the controller is called up (OB251), all variables (DW18 to DW 48 ) with the exception of K, R, TI, TD, BGOG, BGUG, $YH_k$ and $W_k$ are reset in the controller DB. The controller is deactivated. |
| 3 | GESCHW | 0<br>1 | Positioning algorithm<br>Correction rate algorithm |
| 4 | HANDART | 0 | When GESCHW=0:<br>Following the transfer to Manual mode, the specified manipulated variable value YA is adjusted exponentially to the manual value in four sampling steps. Additional manual values are then forwarded immediately to the controller output.<br>When GESCHW=1:<br>The manual values are forwarded immediately to the controller output. The limiting values are in force in Manual mode. |
| | | 1 | When GESCHW=0:<br>The manipulated variable last output is retained.<br>When GESCHW=1:<br>Correction increment $dY_K$ is set to zero. |
| 5 | NO Z | 0<br>1 | With feedforward control<br>No feedforward control |
| 6 and 7 | - | | These bits are not assigned. |
| 8 to 15 | - | | The PID algorithm uses these bits as auxiliary flags. |

The control program can be supplied with fixed values or parameters. Parameters are input via the assigned data words. The controller is based on a PID algorithm. Its output signal can be either a manipulated variable (positioning algorithm) or a manipulated variable modification (correction rate algorithm).

**Correction Rate Algorithm**

The relevant correction increment $dY_k$ is computed at instant t= k · TA according to the following formula:

- Without feedforward control (D11.5=1); XW is forwarded to the differentiator (D11.1=0)

$dY_k = K[(XW_k - XW_{k-1}) R+TI \cdot XW_k+ (TD (XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})]$

$= K (dPW_kR + dI_k + dD_k)$

- With feedforward control (D11.5=0); XW is forwarded to the differentiator (D11.1=0)

$dY_k = K[(XW_k - XW_{k-1}) R+TI \cdot XW_k+ (TD (XW_k - 2XW_{k-1} + XW_{k-2}) + dD_{k-1})]+(Z_k-Z_{k-1})$

$= K (dPW_kR + dI_k + dD_k)+dZ_k$

- Without feedforward control (D11.5=1); XZ is forwarded to the differentiator (D11.1=1)

$dY_k = K[(XW_k - XW_{k-1}) R+TI \cdot XW_k+ (TD (XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})]$

$= K (dPW_kR + dI_k + dD_k)$

- With feedforward control (D11.5=0); XZ is forwarded to the differentiator (D11.1=1)

$dY_k = K[(XW_k - XW_{k-1}) R+TI \cdot XW_k+ (TD (XZ_k - 2XZ_{k-1} + XZ_{k-2}) + dD_{k-1})]+(Z_k-Z_{k-1})$

$= K (dPW_kR + dI_k + dD_k)+dZ_k$

P element I element D element Z element k: $k^{th}$ element

| | | | |
|---|---|---|---|
| When $XW_k$ is applied: | $XW_k$ | = | $W_k - X_k$ |
| | $PW_k$ | = | $XW_k - XW_{k-1}$ |
| | $QW_k$ | = | $PW_k - PW_{k-1}$ |
| | | = | $XW_k-2XW_{k-1}+XW_{k-2}$ |
| When XZ is applied: | $PZ_k$ | = | $XZ_k - XZ_{k-1}$ |
| | $QZ_k$ | = | $PZ_k - PZ_{k-1}$ |
| | | = | $XZ_k-2XZ_{k-1}+XZ_{k-2}$ |
| The result is: | $dPW_k$ | = | $(XW_k- XW_{k-1})R$ |
| | $dI_k$ | = | $TI \cdot XW_k$ |
| | $dD_k$ | = | $(TD \cdot QW_k+dD_{k-1})$ when XW is applied |
| | | = | $(TD \cdot QZ_k+dD_{k-1})$ when XZ is applied |
| | $dZ_k$ | = | $Z_k - Z_{k-1}$ |

**Positioning Algorithm**

The formula used to compute the correction rate algorithm is also used to compute the positioning algorithm.

In contrast to the correction rate algorithm, however, the sum of all correction increments computed (in DW 48), rather than the correction increment $dY_k$ is output at sampling instant $t_k$.

At instant $t_k$ , manipulated variable $Y_k$ is computed as follows:

$$Y_k = \sum_{m=0}^{m=k} dY_m$$

**Initializing the PID Algorithm**

OB251's interface to its environment is the controller DB. All data needed to compute the next manipulated variable value is stored in this DB. Each controller has its own controller data block. The controller-specific data are initialized in a data block that must comprise at least 49 data words. The CPU goes to STOP with a transfer error (TRAF) if no DB is open or if the DB is too short.

**Caution**

Make sure that the right controller DB is open before calling control algorithm OB251.

**Table 9-8. Structure of the Controller DB**

| Data Word | Name | Comments |
|---|---|---|
| 1 | K | Proportional coefficient (-32 768 to + 32 767) for controllers without a derivative-action element<br>Proportional coefficient (- 1500 to +1500) for controllers with a derivative-action element*<br>K is greater than zero when the control is direct acting, and less than zero when the control is reverse acting; the specified value is multiplied by a factor of 0.001**. |
| 3 | R | R parameter (- 32 768 to + 32 767) for controllers without aderivative-action element<br>R parameter (- 1500 to + 1500) for controllers with a derivative-action element*<br>Normally 1 for controllers with P element; the specified value is multiplied with a factor of 0.001** |
| 5 | TI | Constant TI (0 to 9999)<br>$TI = \frac{\text{Sampling interval TA}}{\text{Integral-action time}}$<br>The specified value is multiplied by a factor of 0.001 |
| 3 | | Constant TD (0 to 999)<br>$TD = \frac{\text{Derivative-action time TV}}{\text{Sampling interval TA}}$ |
| 9 | W | Setpoint (- 2047 to +2047) |
| 11 | STEU | Control word (bit pattern) |
| 12 | YH | Value for Manual operation (- 2047 to +2047) |
| 14 | BGOG | Upper limit value (- 2047 to +2047) |
| 16 | BGUG | Lower limit value (- 2047 to +2047) |

* It is possible to have larger gains, if sudden incremental changes to the system deviation are small enough. This is the reason you have to divide larger deviations into smaller ones such as adding the setpoint via a ramp function.

** The factor 0.001 is an approximate value. The exact value of the factor is 1/1024 or 0.000976.

**Table 9-8. Structure of the Controller DB (continued)**

| Data Word | Name | Comments |
|---|---|---|
| 22 | X | Actual value (- 2047 to +2047) |
| 24 | Z | Disturbance variable (- 2047 to +2047) |
| 29 | XZ | Derivative time (- 2047 to +2047) |
| 48 | YA | Output variable (- 2047 to +2047) |

All parameters (with the exception of the control word STEU) must be specified as 16-bit fixed point numbers.

**Caution**

The PID algorithm uses the data words that are not listed in Table 9-8 as auxiliary flags.

### Initialization and Call Up of the PID Controller in a STEP 5 Program

Several different PID controllers can be implemented by calling up OB251 repeatedly. A data block must be initialized prior to each OB251 call up. These DBs serve as data interface between the controllers and the user.

> **Note**
>
> Important controller data are stored in the high-order byte of control word DW11 (DL11). Therefore make sure that only T DR 11/SU D11.0 to D11.7 or RU D 11.0 to D11.7 operations are used to modify user-specific bits in the control word.

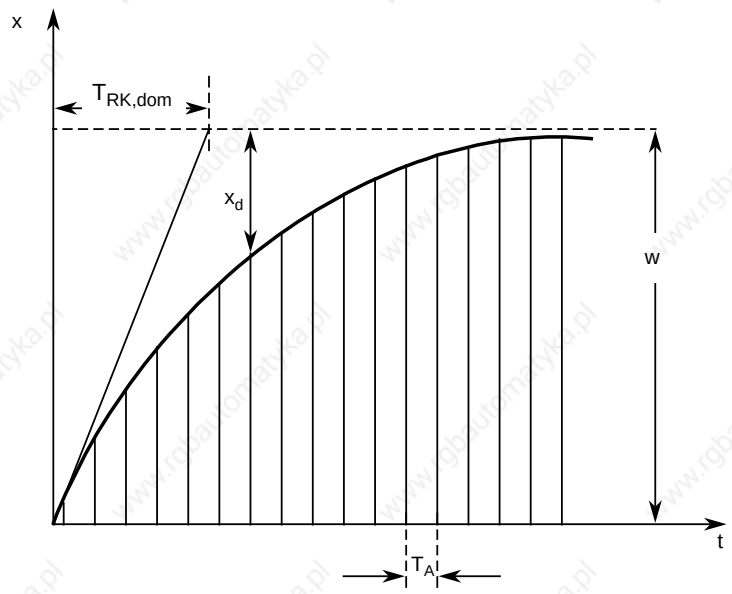
### Selecting the Sampling Interval

In order to be able to use the known analog method of consideration for digital control loops, do not select a sampling interval that is too large.

Experience has shown that a TA sampling interval of approximately 1/10 of the time constant $T_{RK,\,dom}$* produces a control result comparable to the equivalent analog result. Dominant system time constant $T_{RK,\,dom}$ determines the step response of the closed control loop.

$$TA = 1/10 \cdot T_{RK,\,dom}$$

In order to ensure the constancy of the sampling interval, OB251 must always be called up in the service routine for time interrupts (OB13).




| | | |
|---|---|---|
| x | = | Control variable |
| t | = | Time |
| $T_A$ | = | Sampling interval |
| $T_{RK,dom}$ | = | Dominant system time constant of the closed control loop |
| w | = | Reference variable / Setpoint |
| $x_d$ | = | Control deviation |

**Figure 9-8. Principle of Interval Sampling**

* $T_{RK,\,dom}$ = dominant system time constant of the closed control loop

**Example for the Use of the PID Controller Algorithm:**

A PID controller is supposed to keep an annealing furnace at a constant temperature.
The temperature setpoint is entered via a potentiometer.
The setpoints and actual values are acquired using an analog input module and forwarded to the controller. The computed manipulated variable is then output via an analog output module.

The controller mode is set in input byte 0 (see control word DW11 in the controller DB).
You must use the well-known controller design procedure to determine how to tune the controller for each controlled system.

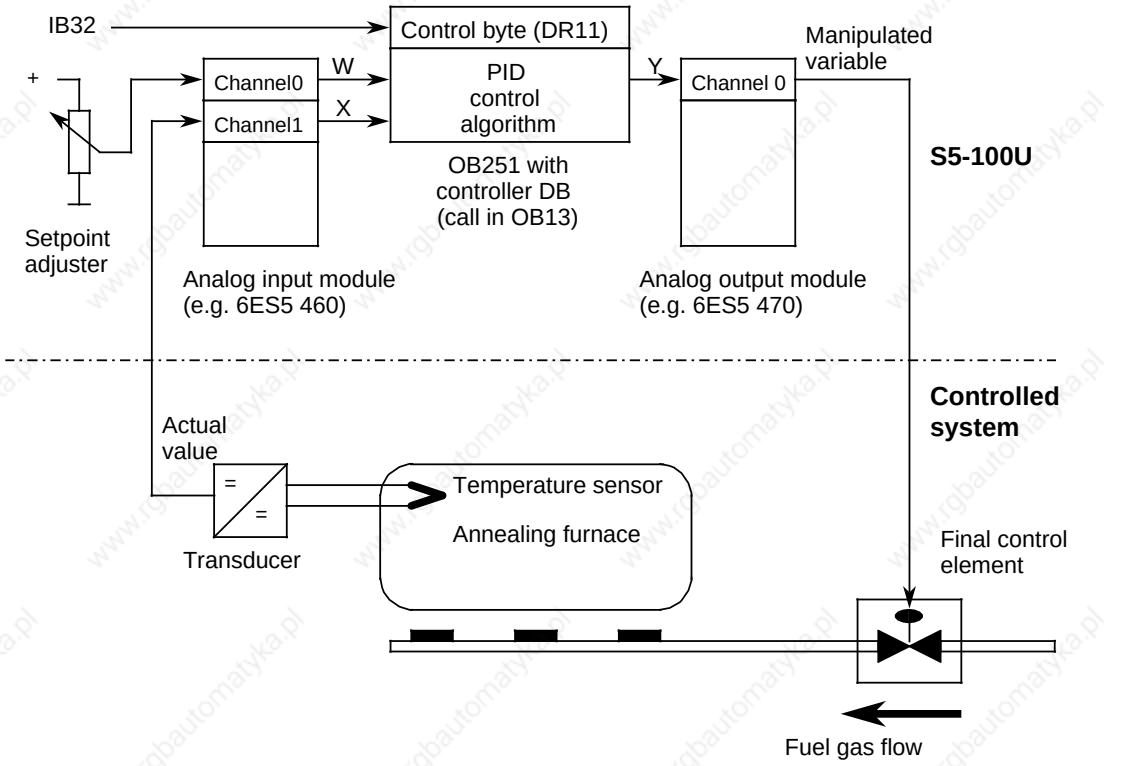



**Figure 9-9. Process Schematic**

The analog signals of the setpoint and actual values are converted into corresponding digital values in each sampling interval (set in OB13). OB251 uses these values to compute the new digital manipulated variable, from which, in turn, the analog output module generates a corresponding analog signal. This signal is then forwarded to the controlled system.

Calling the Controller in the Program:

| OB 13 STL | Description |
|---|---|
| :<br>:JU FB 10<br>NAME :CONTROLLER 1<br>:<br>:<br>:<br>:<br>:<br>:<br>:<br>:<br>:BE | PROCESS CONTROLLER<br><br>THE CONTROLLER'S SAMPLING INTERVAL<br>DEPENDS ON THE TIME BASE USED<br>TO CALL OB13 (SET IN DB1).<br>THE DECODING TIME OF THE ONBOARD<br>ANALOG INPUTS MUST BE TAKEN<br>INTO ACCOUNT WHEN SELECTING<br>THE SAMPLING INTERVAL. |

| FB10 | STL | Description |
|---|---|---|
| NAME | :CONTROLLER 1 | |
| | : | |
| | :C DB 30 | SELECT CONTROLLER'S DB |
| | : | |
| | : | ********** |
| | : | READ CONTROLLER'S CONTROL BITS |
| | : | ********** |
| | : | |
| | :L PY 0 | READ CONTROLLER'S |
| | :T FY 10 | CONTROL BITS |
| | :T DR 11 | AND STORE IN DR11 |
| | : | NOTE CAREFULLY: |
| | : | DR11 CONTAINS IMPORTANT CONTROL |
| | : | DATA FOR OB251 |
| | : | THE CONTROL BITS MUST |
| | : | THEREFORE BE TRANSFERRED WITH |
| | : | T DR11 TO PREVENT |
| | : | CORRUPTING DL11 |
| | : | |
| | : | ********** |
| | : | READ ACTUAL VALUE AND SETPOINT |
| | : | ********** |
| | : | |
| | :A F 12.0 | FLAG 0 (FOR UNUSED FUNCTIONS |
| | :R F 12.0 | IN FB 250) |
| | :AN F 12.1 | FLAG 1 |
| | :S F 12.1 | |
| | : | |
| | :JU FB250 | READ ACTUAL VALUE |
| NAME | :RLG: AI | |
| BG | : KF +8 | MODULE ADDRESS |
| KNKT | : KY 0,6 | CHANNEL NO. 0, FIXED-POINT BIPOLAR |
| OGR | : KF +2047 | UPPER LIMIT FOR ACTUAL VALUE |
| UGR | : KF - 2047 | LOWER LIMIT FOR ACTUAL VALUE |
| EINZ | : F 12.0 | NO SELECTIVE SAMPLING |
| XA | : DW 22 | STORE SCALED ACTUAL VAL. IN CONTR. DB |
| FB | : F 12.2 | ERROR BIT |
| BU | : F 12.3 | RANGE VIOLATION |
| | : | |

| FB10 (continued) STL | Explanation |
|---|---|
| : | |
| :JU FB250 | READ SETPOINT |
| NAME :RLG: AI | |
| BG : KF +8 | MODULE ADDRESS |
| KNKT : KY 1,6 | CHANNEL NO. 1, FIXED-POINT BIPOLAR |
| OGR : KF +2047 | UPPER LIMIT FOR SETPOINT |
| UGR : KF - 2047 | LOWER LIMIT FOR SETPOINT |
| EINZ : F 12.0 | NO SELECTIVE SAMPLING |
| XA : DW 9 | STORE SCALED SETPOINT IN CONTR. DB |
| FB : F 13.1 | ERROR BIT |
| BU : F 13.2 | RANGE VIOLATION |
| : | |
| :A F 10.0 | IN MANUAL MODE, THE SETPOINT IS |
| :JC =WEIT | SET TO THE ACTUAL VALUE TO FORCE |
| :L DW 22 | THE CONTROLLER TO REACT |
| :T DW 9 | TO A SYSTEM DEVIATION, IF ANY, |
| : | WITH A P STEP |
| : | ON TRANSFER |
| : | TO AUTOMATIC |
| : | MODE |
| WEIT : | |
| : | ********** |
| : JU OB251 | CALL CONTROLLER |
| : | ********** |
| : | |
| : | ********** |
| : | OUTPUT MANIPULATED VALUE |
| : | ********** |
| :JU FB251 | |
| NAME :RLG:AQ | |
| XE : DW 48 | |
| BG : KF +8 | MODULE ADDRESS |
| KNKT : KY 0,1 | CHANNEL 0, FIXED-POINT BIPOLAR |
| OGR : KF +2047 | UPPER LIMIT FOR ACTUATING SIGNAL |
| UGR : KF - 2047 | LOWER LIMIT FOR ACTUATING SIGNAL |
| FEH : F 13.5 | ERROR BIT WHEN LIMITING VAL. DEFINED |
| BU : F 13.6 | MANIPULATED VARIABLE Y TO ANALOG |
| : | OUTPUT |
| :BE | RANGE VIOLATION |

| DB 30 | STL | Explanation |
|---|---|---|
| 0: | KH = 0000; | |
| 1: | KF = +01000; | K PARAMETER (HERE=1), FACTOR 0.001 |
| 2: | KH = 0000; | (VALUE RANGE: - 32768 TO 32767) |
| 3: | KF = +01000; | R PARAMETER (HERE=1), FACTOR 0.001 |
| 4: | KH = 0000; | (VALUE RANGE: - 32768 TO 32767) |
| 5: | KF = +00010; | TI=TA/TN (HERE=0.01), FACTOR 0.001 |
| 6: | KH = 0000; | (VALUE RANGE: 0 TO 9999) |
| 7: | KF = +00010; | TD=TV/TA (HERE=10), FACTOR 1 |
| 8: | KH = 0000; | (VALUE RANGE: 0 TO 999) |
| 9: | KF = +00000; | SETPOINT W, FACTOR 1 |
| 10: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 11: | KM = 00000000 00100000; | CONTROL WORD |
| 12: | KF = +00500; | MANUAL VALUE YH, FACTOR 1 |
| 13: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 14: | KF = +02000; | UPPER CONT. LIMIT BGOG, FACTOR 1 |
| 15: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 16: | KF = -02000; | LOWER CONT. LIMIT BGUG, FACTOR 1 |
| 17: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 18: | KH = 0000; | |
| 19: | KH = 0000; | |
| 20: | KH = 0000; | |
| 21: | KH = 0000; | |
| 22: | KF = +00000; | ACTUAL VALUE X, FACTOR 1 |
| 23: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 24: | KF = +00000; | DISTURBANCE VARIABLE Z, FACTOR 1 |
| 25: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 26: | KH = 0000; | |
| 27: | KH = 0000; | |
| 28: | KH = 0000; | |
| 29: | KF = +00000; | FEEDFORWARD XZ FOR DIFF., |
| 30: | KH = 0000; | FACTOR 1, (- 2047 TO 2047) |
| 31: | KH = 0000; | |
| 32: | KH = 0000; | |
| 33: | KH = 0000; | |
| 34: | KH = 0000; | |
| 35: | KH = 0000; | |
| 36: | KH = 0000; | |
| 37: | KH = 0000; | |
| 38: | KH = 0000; | |
| 39: | KH = 0000; | |
| 40: | KH = 0000; | |
| 41: | KH = 0000; | |
| 42: | KH = 0000; | |
| 43: | KH = 0000; | |
| 44: | KH = 0000; | |
| 45: | KH = 0000; | |
| 46: | KH = 0000; | |
| 47: | KH = 0000; | |
| 48: | KF = +00000; | CONTROLLER OUTPUT Y, FACTOR 1 |
| 49: | KH = 0000; | (VALUE RANGE: - 2047 TO 2047) |
| 50: | | |

## 10 Interrupt Processing


10.1 Interrupt Processing with OB2, for CPU 103 Version 8MA02 and Higher . . . . . 10 - 1

10.2 Calculating Interrupt Reaction Times . . . . . 10 - 5

**Figures**


10-1 Possible Configuration of the Programmable Controller with Bus Units Having Interrupt Capability . . . . . 10 - 1
10-2 Program Interruptions by Process Interrupts . . . . . 10 - 2
10-3 Accessing the Process Image Tables from OB2 . . . . . 10 - 4


**Tables**


10-1 Additional Reaction Times . . . . . 10 - 5

# 10 Interrupt Processing, for CPU 103 Version 8MA02 and Higher

Interrupt-driven program processing starts when a signal from the CPU causes the programmable controller to interrupt cyclic or time-controlled program scanning in order to process a specific program. Once this program has been scanned, the CPU returns to the point of interruption in the cyclic or time-controlled program and resumes processing at that point.

**Prerequisites for Interrupt-Driven Program Processing**

Interrupt-driven program processing is possible only if the following conditions are met:

- The bus unit with interrupt capability is directly adjacent to the CPU (slots 0 and 1).
- Four-channel digital input modules or comparator modules must be mounted on the bus unit to transfer process interrupts.
  - You may plug other modules in, but these modules will have no interrupt handling capability.
- The programmable controller is in the Power ON state and in the RUN operating mode.
- Interrupt processing is not disabled by an IA operation in your program. See section 8.2.8.
- OB2 has been programmed.

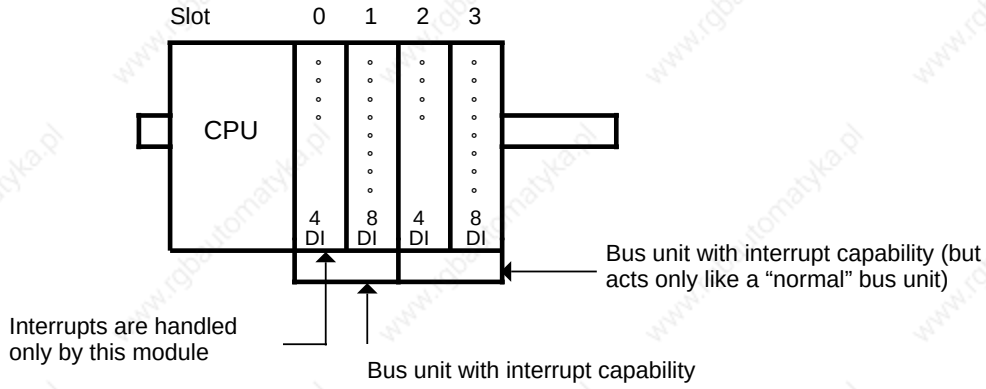



**Figure 10-1. Possible Configuration of the Programmable Controller with Bus Units Having Interrupt Capability**

## 10.1 Interrupt Processing with OB2, for CPU 103 Version 8MA02 and Higher

For interrupt-driven processing, OB2 must have been programmed. OB2 is called up by a process interrupt and interrupts in turn the cyclic or time-controlled program scanning. Other blocks can be called from OB2. After the interrupt-driven program has been processed, the CPU resumes cyclic or time-controlled program scanning.

## Triggering an Interrupt

Interrupts can only be triggered by four-channel digital input modules and comparator modules that are plugged into slots 0 and 1 on a bus unit with interrupt capability.

Interrupts are triggered by a change in the signal state (0 1=positive edge; 1 0=negative edge) at the respective interrupt input. Then the programmable controller automatically branches to OB2. If you have not programmed OB2, either the cyclic or time-controlled program resumes immediately after the interrupt.

The cyclically processed program can be interrupted after every STEP 5 statement.

The processing of integral FBs can be interrupted at certain points (see section 9.2). The data cycle (see section 2.2.2) can be interrupted after each data packet consisting of four data bits and a check bit.

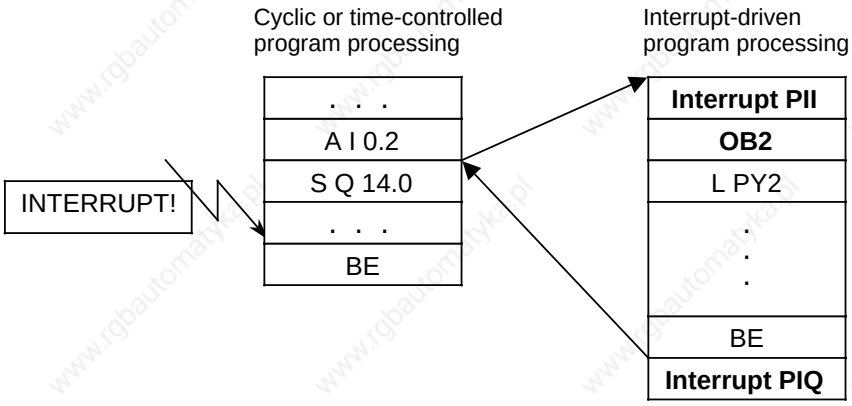



**Figure 10-2. Program Interruptions by Process Interrupts**

Use the IA command to disable interrupt processing. Use the RA command to enable interrupt processing. The default setting is RA (see section 8.2.8).

> **Note**
>
> Even for interrupt processing, you may not exceed the general block nesting depth of 16 levels.

## Interrupt Priorities

If a second interrupt is triggered during an interrupt processing, the second interrupts is processed at the end of the first interrupt processing.

> **Note**
>
> If both a positive and negative pulse edge occur at an interrupt input while the IA operation is valid (disable interrupt), it is no longer possible to determine the channel that has triggered the interrupt.
> But after an RA operation, OB2 is still called up.

## Reading Out the Interrupt PII

If a process interrupt occurs, only the signal states of the interrupt inputs in slots 0 and 1 are read out to the interrupt PII.

This data in the interrupt PII is the only data provided to the interrupt-driven program for evaluation.

The interrupt PII can be scanned in OB2 by means of the following load operations:

**Overview:**

| Operation | Operand | Description |
|---|---|---|
| L | PY 0 | Load byte 0 of the interrupt PII into ACCU 1 |
| L | PY 1 | Load byte 1 of the interrupt PII into ACCU 1 |
| L | PW 0 | Load word 0 of the interrupt PII into ACCU 1 |

If you enter other parameters, the CPU goes into the STOP mode and enters the “NNN” error message in the ISTACK (see section 5.2). When data is read into the interrupt PII, the normal PII is not written to simultaneously.

## Writing to the Interrupt PIQ

Data from time-controlled or interrupt-driven programs to I/O modules are written to the interrupt PIQ and simultaneously to the “normal” PIQ.

After OB2 is finished, the data that has been transferred to the interrupt PIQ is output to the peripheral I/Os in an interrupt output data cycle (before “normal” program processing).

After the OB1 program cycle, the PIQ is copied to the interrupt PIQ.

The interrupt output data cycle is executed only if the interrupt PIQ has been written to. Use transfer statements to write data for I/O modules to the interrupt PIQ. When data is written to the interrupt PIQ, data is written simultaneously to the normal PIQ.

**Overview:**

| Operation | Operand | Description |
|---|---|---|
| T | PY 0 to 127 | Transfer contents of ACCU 1 into the interrupt PIQ |
| T | PW 0 to 126 | Transfer contents of ACCU 1 into the interrupt PIQ |

## Possibilities of Accessing Process I/O Image Tables

The following figure shows how data transfer between the process I/O image tables and ACCU 1 takes place when using various load and transfer statements in OB2.

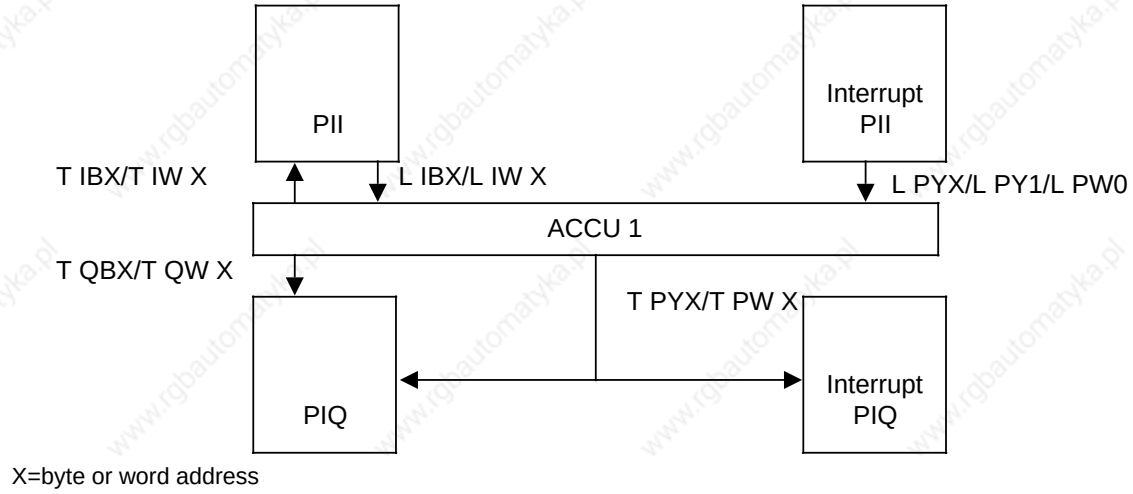



**Figure 10-3. Accessing the Process Image Tables from OB2**

## Example of How OB2 Can Be Programmed

Binary statements can access only the normal PII and PIQ. In order to determine the channel that triggered an interrupt, transfer the I/O byte or word, for example, to a flag byte or word and then evaluate using binary statements.

| Example | STL OB2 | Explanation |
|---|---|---|
| Two sensors are connected on channels 0 and 1 on a four-channel digital input module on slot 0.<br>Branch to FB12 if sensor 1 (channel 0) triggers an interrupt. | `L PY 0`<br>`T FY 0`<br>`A F 0.0`<br>`AN I 0.0`<br>`O`<br>`AN F 0.0`<br>`A I 0.0`<br>`JC FB 12`<br>`...` | Load byte 0 of the interrupt PII into ACCU 1 and transfer it to flag byte 0.<br>Did a positive edge occur on channel 0?<br>OR<br>Did a negative edge occur on channel 0?<br>If a pulse edge has occurred, a branch is made to FB12. |

### Caution

Be sure to save the flags (in a data block, for example) if these flags are to be overwritten during interrupt processing and are needed again in the cycle.

## 10.2 Calculating Interrupt Reaction Times

The total reaction time is is the sum of the following times:

- Signal delay of the module triggering the interrupt (= time from the input signal change triggering the interrupt to the activation of the interrupt line)
- Interrupt reaction time of the CPU
- Execution time of the interrupt program (= sum of all STEP 5 statements in the interrupt evaluation program)

Calculate the interrupt reaction times as follows:

CPU's interrupt reaction time = basic reaction time + additional reaction times

The basic reaction time is 0.6 ms and is valid if the following conditions exist:

- No integrated FBs were used.
- No parameters for the integral clock are set.
- No programmer/OP functions are present.
- OB13 has not been programmed.
- No SINEC L1 is connected.

The additional reaction times are variable. They are listed in Table 10-1.

**Table 10-1. Additional Reaction Times**

| Additional Running Functions of the Programmable Controller | Interrupt Reaction Times |
|---|---|
| Integrated FBs | 0.5 ms |
| Parameters set for clock | 0.2 ms |
| SINEC L1 bus to the programmer interface | 8.0 ms |
| OP functions | Depending on the number of bytes used for loading the memory |
| Programmer functions:<br><br>Status block/Transfer block<br>Output address<br><br>Compress<br>• If no blocks are moved<br><br>• If blocks are moved | <br><br>0.5 ms<br>18 ms per kbyte<br><br><br>• Depending on the number of blocks present (after overall reset 31 ms)<br><br>• 600 ms per each 1kword of instructions in the block to be moved |

## 11 Analog Value Processing


| | | |
|---|---|---|
| 11.1 | Analog Input Modules | 11 - 1 |
| 11.2 | Connecting Current and Voltage Sensors to Analog Input Modules | 11 - 1 |
| 11.2.1 | Voltage Measurement with Isolated/Non-Isolated Thermocouples | 11 - 2 |
| 11.2.2 | Two-Wire Connection of Voltage Sensors | 11 - 3 |
| 11.2.3 | Two-Wire Connection of Current Sensors | 11 - 4 |
| 11.2.4 | Connection of Two-Wire and Four-Wire Transducers | 11 - 4 |
| 11.2.5 | Connection of Resistance Thermometers | 11 - 6 |
| 11.3 | Start-Up of Analog Input Modules | 11 - 7 |
| 11.4 | Analog Value Representation of Analog Input Modules | 11 - 11 |
| 11.5 | Analog Output Modules | 11 - 19 |
| 11.5.1 | Connection of Loads to Analog Output Modules | 11 - 19 |
| 11.5.2 | Analog Value Representation of Analog Output Modules | 11 - 20 |
| 11.6 | Analog Value Conversion: Function Blocks FB250 and FB251 | 11 - 22 |
| 11.6.1 | Reading in and Scaling an Analog Value -FB250- | 11 - 22 |
| 11.6.2 | Output of Analog Value -FB251- | 11 - 25 |

**Figures**


11-1 Voltage Measuring with Isolated Thermocouples (6ES5 464-8MA11/8MA21) . . . . . 11 - 2
11-2 Voltage Measuring with Non-Isolated Thermocouples (6ES5 464-8MA11/8MA21) . . . . . 11 - 2
11-3 Two-Wire Connection of Voltage Sensors (6ES5 464-8MB11, 464-8MC11, 466-8MC11) . . . . . 11 - 3
11-4 Two-Wire Connection for Current Sensors (6ES5 464-8MD11) . . . . . 11 - 4
11-5 Connection of Two-Wire Transducers (6ES5 464-8ME11) . . . . . 11 - 4
11-6 Connection for Four-Wire Transducers (6ES5 464-8ME11) . . . . . 11 - 5
11-7 Wiring Method for PT 100 (6ES5 464-8MF11/8MF21) . . . . . 11 - 6
11-8 Wiring Possibilities for Input Modules (6ES5 464-8MF11) . . . . . 11 - 6
11-9 Load Connection via a Four-Wire Circuit (6ES5 470-8MA11, 6ES5 470-8MD11) . . . . . 11 - 19
11-10 Load Connection via a Two-Wire Circuit (6ES5 470-8MB11, 6ES5 470-8MC11) . . . . . 11 - 20
11-11 Scaling Schematic for FB250 . . . . . 11 - 22
11-12 Schematic for “Display of Tank Make-Up Quantity” . . . . . 11 - 23
11-13 Conversion of the Nominal Range into the Defined Range . . . . . 11 - 23
11-14 Schematic for “Display of Tank Contents” . . . . . 11 - 25
11-15 Transformation of the Analog Value to the Nominal Range . . . . . 11 - 26

**Tables**


| | | |
|---|---|---|
| 11-1 | Operating Mode Switch Settings for Analog Input Modules 464-8 to 11 | 11 - 7 |
| 11-2 | Operating Mode Switch Settings for Analog Input Module 464-8MA21 | 11 - 8 |
| 11-3 | Operating Mode Switch Settings for Analog Input Module 464-8MF21 | 11 - 10 |
| 11-4 | Representation of an Analog Input Value as Bit Pattern | 11 - 11 |
| 11-5 | Analog Input Module 464-8MA11, -8MF11, -8MB11 (Bipolar Fixed-Point Number) | 11 - 11 |
| 11-6 | Analog Input Module 464-8MC11, -8MD11 (Bipolar Fixed-Point Number) | 11 - 12 |
| 11-7 | Analog Input Module 464-8ME11, 4x4 to20 mA (Absolute Value Representation) | 11 - 12 |
| 11-8 | Analog Input Module 464-8MF11, 2x PT 100 (Unipolar) Analog Input Module 464-8MF21, 2x PT 100 “No Linearization” (Unipolar) | 11 - 12 |
| 11-9 | Analog Input Module 464-8MF21, 2x PT 100 “with Linearization” (Bipolar) | 11 - 13 |
| 11-10 | Analog Input Module 464-8MA21, 4x±50 mV “with Linearization” and “with Temperature Compensation” (Bipolar); Thermoelement Type K (Nickel-Chromium/Nickel-Aluminium, according to IEC 584) | 11 - 14 |
| 11-11 | Analog Input Module 464-8MA21, 4x±50 mV “with Linearization” and “with Temperature Compensation” (Bipolar); Thermoelement Type J (Iron/Copper-Nickel (Konstantan), according to IEC 584) | 11 - 15 |
| 11-12 | Analog Input Module 464-8MA21, 4x±50 mV “with Linearization” and “with Temperature Compensation” (Bipolar); Thermoelement Type L (Iron/Copper-Nickel (Konstantan) according to DIN 43710) | 11 - 16 |
| 11-13 | Analog Input Module 466-8MC11, 4x 0 to10 V | 11 - 16 |
| 11-14 | Representation of an Analog Output Value as a Bit Pattern | 11 - 20 |
| 11-15 | Output Voltages and Currents for Analog Output Modules (Fixed-Point Number Bipolar) | 11 - 21 |
| 11-16 | Output Voltages and Currents for Analog Output Modules (Unipolar) | 11 - 21 |
| 11-17 | Call and Parameter Assignments of FB250 | 11 - 22 |
| 11-18 | Call and Parameter Assignments of FB251 | 11 - 25 |

# 11 Analog Value Processing

## 11.1 Analog Input Modules

Analog input modules convert analog process signals to digital values that the CPU can process (via the process image input table, PII). In the following sections, you will find information about the operating principle, wiring methods, and start-up and programming of analog input modules.

## 11.2 Connecting Current and Voltage Sensors to Analog Input Modules

Observe the following rules to connect current and voltage sensors to analog input modules:

- When you have multi-channel operations, assign the channels in ascending order. This shortens the data cycle.
- Use terminals 1 and 2 for the connection of a compensating box (464-8MA11 ) or for the supply of two-wire transducers (464-8ME11).
  - Terminals 1 and 2 cannot be used with the remaining analog input modules.
- Short-circuit the terminals of unused inputs.
- Set the reference potentials of the sensors to a common reference potential. Do this to prevent the potential difference between the common references from exceeding 1 V.

### 11.2.1 Voltage Measurement with Isolated/Non-Isolated Thermocouples

Module **464-8MA11/8MA21** is recommended for voltage measurement with thermocouples. With **floating sensors** (e. g., isolated thermocouples), the permissible potential difference $V_{CM}$ between terminals of the inputs and the potential of the standard mounting rail must not be exceeded. To avoid this, the negative potential of the sensor must be connected to the central ground point (see Figure 11-1). Jumper terminals 1 and 2 together if you do not use compensation boxes.

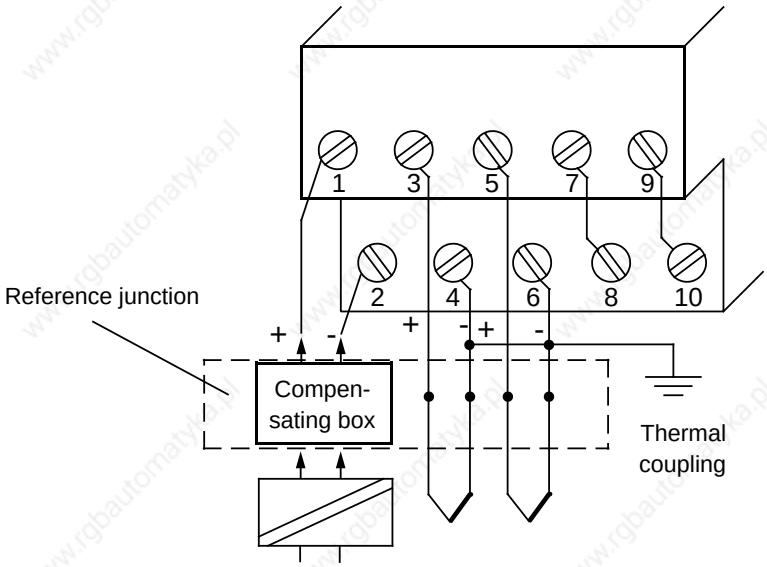



**Figure 11-1. Voltage Measuring with Isolated Thermocouples (6ES5 464-8MA11/8MA21)**

With **non-floating sensors** (e. g., non-isolated thermocouples), the permissible potential difference $V_{CM}$ must not be exceeded (see maximum values of the individual modules).

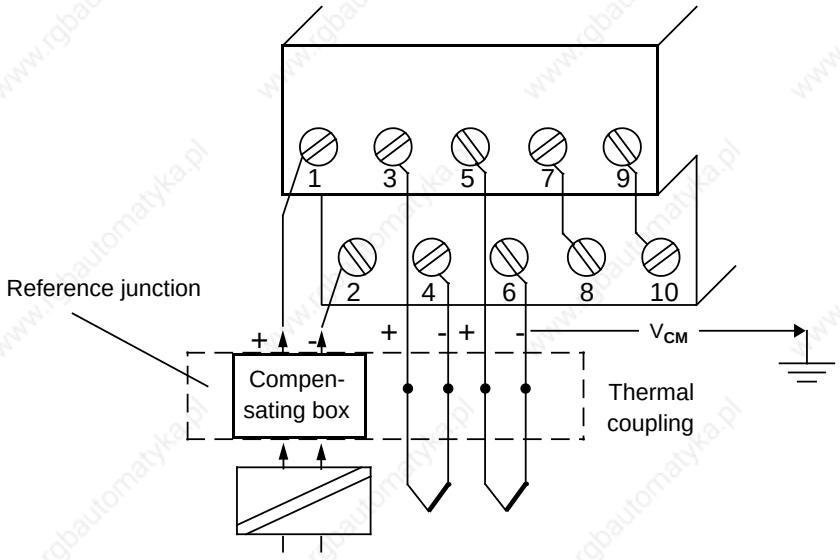



**Figure 11-2. Voltage Measuring with Non-Isolated Thermocouples (Module 6ES5 464-8MA11/8MA21)**

**Connection of Thermocouples with Compensating Box to Module 464-8MA11/8MA21**

The influence of the temperature on the reference junction (e. g., terminal box) can be compensated for with a compensation box. Observe the following rules:

- The compensation box must have a floating supply.
- The power supply must have a grounded shielding winding.
- The compensation box must be connected to terminals 1 and 2 of the terminal block.

## 11.2.2 Two-Wire Connection of Voltage Sensors

You can use the following three modules for the connection of voltage sensors:

- Analog Input Module **464-8MB11** for voltages of±1 V
- Analog Input Module **464-8MC11** for voltages of±10 V
- Analog Input Module **466-8MC11** for voltages from 0 to 10 V

Figure 11-3 shows the two-wire connection of voltage sensors.

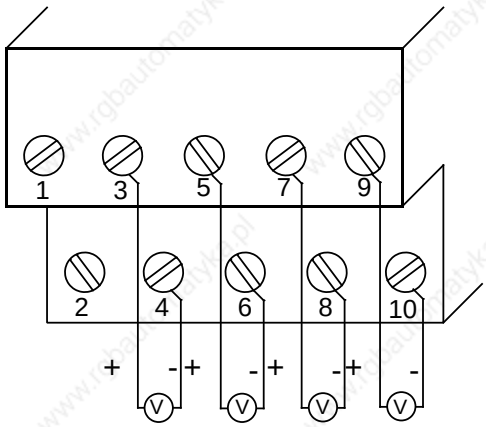



**Figure 11-3. Two-Wire Connection of Voltage Sensors (6ES5 464-8MB11, 464-8MC11, 466-8MC11)**

### 11.2.3 Two-Wire Connection of Current Sensors

You can use module **464-8MD11** for the two-wire connection of current sensors.
Figure 11-4 shows the two-wire connections of current sensors.

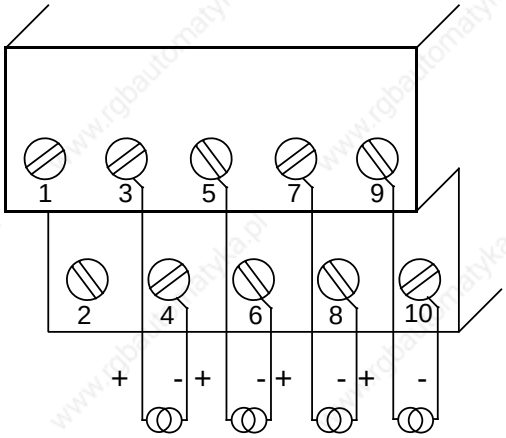



**Figure 11-4. Two-Wire Connection for Current Sensors (6ES5 464-8MD11)**

### 11.2.4 Connection of Two-Wire and Four-Wire Transducers

Use the 24-V inputs 1 and 2 of analog input module **464-8ME11** to supply the two-wire transducers. The two-wire transducer converts the supplied voltage to a current of 4 to 20 mA.
For wiring connections, see Figure 11-5.

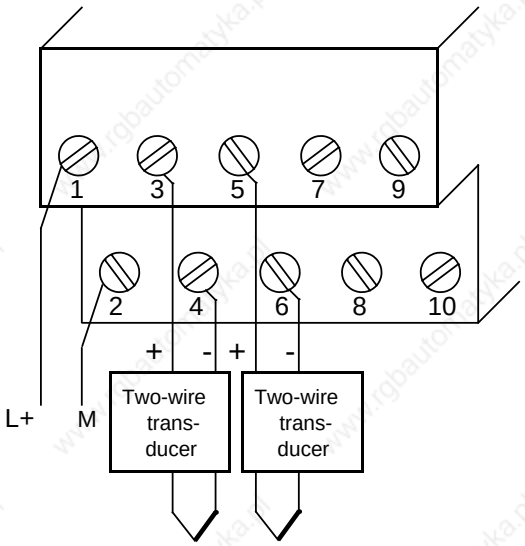



**Figure 11-5. Connection of Two-Wire Transducers (6ES5 464-8ME11)**

If you use a four-wire transducer connect it as shown in Figure 11-6.

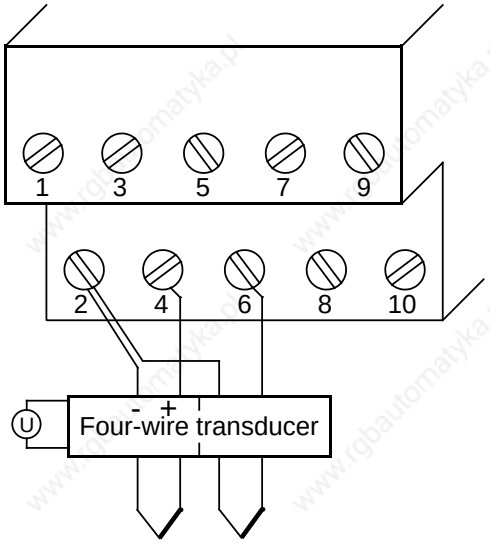


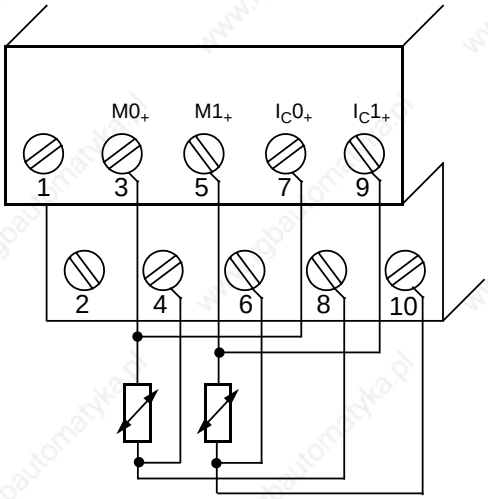

**Figure 11-6. Connection for Four-Wire Transducers (6ES5 464-8ME11)**

Four-wire transducers require their own power supply. Connect the “+” pole of the four-wire transducer to the corresponding “-” pole of the terminal block (a connection technique that is the opposite of the two-wire transducer). Connect negative terminals of the four-wire transducer to terminal two of the terminal block.

Inputs 4, 6, 8, and 10 of the analog input module **464-8ME11** are connected internally via shunt resistors. Because of the internal shunt resistors, broken wire signaling is not possible.

## 11.2.5 Connection of Resistance Thermometers

Analog input module **464-8MF11/8MF21** is suited for the connection of resistance thermometers (e.g., PT 100).
The resistance of the PT 100 is measured in a four-wire circuit. A constant current is supplied to the resistance thermometer via terminals 7 and 8 as well as via terminals 9 and 10, so that voltage drops in these “constant current circuits” do not affect the measurement results. The measuring inputs have a high resistance so that only a negligible current loss develops in the measuring circuits.




Terminal assignments:

| | |
|---|---|
| (3/4): | Measuring circuit M0 |
| (5/6): | Measuring circuits M1 |
| (7/8): | Constant current circuit $I_C$ 0 |
| (9/10): | Constant current circuit $I_C$ 1 |

**Figure 11-7. Wiring Method for PT 100 (6ES5 464-8MF11/8MF21)**

If you use only one channel for PT 100 measurement (e.g., channel 0), then you can use the other channel for voltage measurement (± 500 mV). In this case, use terminals M+/M- for the signal connection and short circuit the terminals $I_{C+}$ and $I_{C-}$.

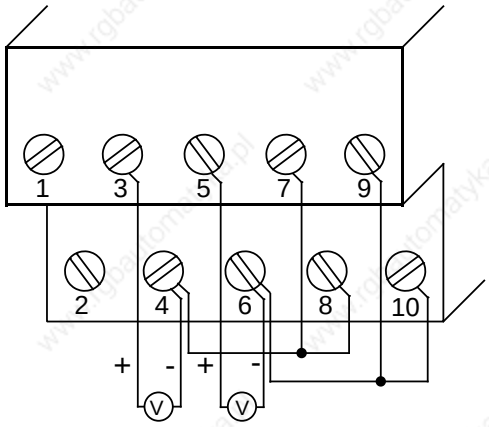



**Figure 11-8. Wiring Possibilities for Input Modules (6ES5 464-8MF11)**

## 11.3 Start-Up of Analog Input Modules

Set the intended operating mode using the switches on the front panel of analog input modules 464-8 through 11. These switches are located on the right side at the top of the front panel of the module.

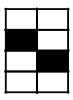
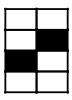
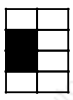
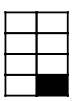
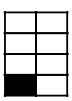
**Power supply frequency:** Set the switch to the available power supply frequency. This selects the integration time of the A/D converters for optimal interference voltage suppression.
Power frequency 50 Hz: Integration time 20 ms
Power frequency 60 Hz: Integration time 16.66 ms

**Operation:** Set the number of channels you wish to assign on the input module. If there are fewer than four channels, less address space will be assigned and measured values will be updated faster.

**Broken wire:** Once the broken wire signal has been activated, a break on one of the lines to the sensor (thermocouple or PT 100) or of the sensor itself causes the red LED above the function selection switch to light up. At the same time, the broken wire error bit F (bit 1, byte 1) for the faulty channel is set.

The module “recognizes” a wire break by applying a conventional tripping current to the input terminals and by comparing the resulting voltage to a limit value. If there is a wire break in the sensor or the lines, the voltage exceeds the limit value and a “wire break” signal is sent. When the signal at the input is measured with a digital voltmeter, the tripping current pulses cause apparent fluctuations of the signal. Deactivation of the wire break signal does **not** turn off the tripping current.

**Table 11-1. Operating Mode Switch Settings for Analog Input Modules 464-8 to 11**

| Function | Settings for Operating Mode Switch | | |
|---|---|---|---|
| Power supply frequency | 50 Hz | 60 Hz | |
| Operation | 1 channel (channel 0)   | 2 channels (channel 0 and channel 1)   | 4 channels (channel 0 - channel 3)   |
| Wire break | With wire break signal   | No wire break signal   | |

Additional operating mode switch selections possible with analog module **464-8MA21**:

**Linearization:** With this function, you can obtain a characteristic linearization of the thermocouples of type J, K, and L or of the resistance thermometer PT 100. With module 464-8MA21, the linearization must always be activated together with the corresponding compensation of the reference point temperature.

**Thermocouples:**
Type J: - 200° C to +1200° C (-328° F to 2192° F)
Type K: - 200° C to +1369° C (-328° F to 2497° F)
Type L: - 199° C to + 900° C (-326° F to 1652° F) in steps each of 1° C (1.8° F)

**Temperature compensation:** For the thermocouples of type J, K, and L, you can compensate, on the one hand, the temperature of the reference point using a compensating box. (See Figure 11-1). On the other hand, it is possible to move the reference point to the front of the module by activating the "temperature compensation" function. When thermocouples are directly connected, an internal circuit on the module causes the digital value "0" to be displayed independently of the temperature of the terminal when the temperature at the measuring junction is 0° C (32° F). In order to accomplish this, the terminals of the sensors have to be connected directly to the module, i.e., without a copper extension cable.

**Table 11-2. Operating Mode Switch Settings for Analog Input Module 464-8MA21**

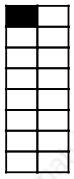
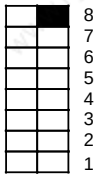
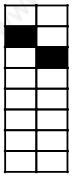
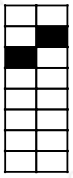
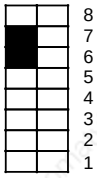
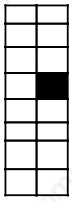
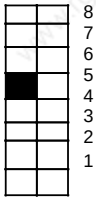
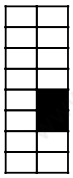
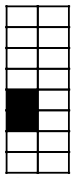
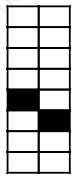
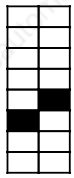
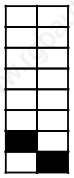
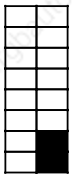
| Function | Settings for Operating Mode Switch | | |
|---|---|---|---|
| Power supply frequency | 50 Hz  | 60 Hz  | |
| Operation | 1 channel (channel 0)  | 2 channels (channel 0 and channel 1)  | 4 channels (channel 0 - channel 3)  |
| Wire break | With wire break signal  | No wire break signal  | |

**Table 11-2. Operating Mode Switch Settings for Analog Input Module 464-8MA21 (continued)**

| Function | Settings for Operating Mode Switch | | | |
|---|---|---|---|---|
| Characteristic linearization of thermocouples | without linearization  | Linearization type K  | Linearization type J  | Linearization type L  |
| Temperature compensation | without temperature compensation | Temperature compensation for type K  | Temperature compensation for types J and L  | |

If you have set “Characteristic linearization” and “Temperature compensation” with the operating mode switches on module **464-8MA21** for the thermocouple used, then the reference temperature is 0° C (32° F). This means that with 0° C (32° F) at the measuring junction, the value “0” is displayed.

If you equip several channels with thermocouples, use the same type of thermocouple. If you select mixed thermocouples, or if you use thermocouples other than type J, K, or L, then you must choose the following two settings:

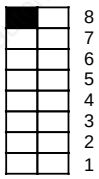
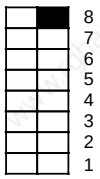
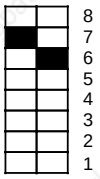
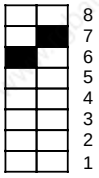
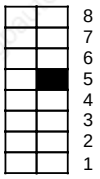
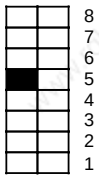
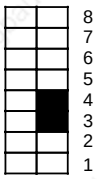
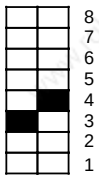
- “No linearization”
- “No temperature compensation”

Compensation is then not possible even with a compensating box because the compensating box is designed only for a certain type of thermocouple. It is possible to use a thermostat in the terminal box if you use the thermostat temperature in the application software to adjust the thermocouple input offset.

When you set the switches to “no linearization” and “no temperature compensation”, then module 464-8MA21 functions just like module 464-8MA11.

Set the switches on analog module **464-8MF21** as illustrated in Table 11-3.

**Table 11-3. Operating Mode Switch Settings for Analog Input Module 464-8MF21**

| Function | Settings for Operating Mode Switch | |
|---|---|---|
| Power supply frequency | 50 Hz  | 60 Hz  |
| Operation | 1 channel (channel 0)  | 2 channels (channel 0 and channel 1)  |
| Wire break | With wire break signal  | No wire break signal  |
| Characteristic linearization for the PT 100 | No linearization  | Linearization for PT 100  |

Position 1 and 2 on the operating mode switch have no function.

If you set the switch to “no linearization” and “no temperature compensation”, module 464-8MF21 functions just like module 464-8MF11.

The characteristic linearization is possible for the following temperature ranges.

**PT 100**: -100° C to+850° C (-148° F to 1569° F) (in steps of 0.5° C (0.9° F))

## 11.4 Analog Value Representation of Analog Input Modules

Each analog process signal has to be converted into a digital format, to be stored in the process image input table (PII). The analog signals are converted into a binary digit that is written in one of the following ways:

- In one byte (466-8MC11)
- In two bytes (the remaining analog input modules)

Each bit position has a fixed value in powers of two (see Tables 11-4 and 11-14). Analog values are represented in two's complement.

The following tables show the analog value representations of the different analog inputs in 2-byte format. You will need this information to program FB250 and FB251 (see section 11.6).

**Table 11-4. Representation of an Analog Input Value as Bit Pattern**

| | High Byte | | | | | | | | Low Byte | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit Number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Analog Value Represent. | S | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ | X | E | OV |

Key:

| | | |
|---|---|---|
| S | Sign bit | 0="+", 1="-" |
| X | Irrelevant bits | |
| E | Error bit | 0= no wire break; 1=wire break |
| OV | Overflow bit | 0= Measured value 4095 units at the most<br>1= Measured value greater than or equal to 4096 units |

**Analog value representation for analog input modules 464-8...**

**Table 11-5. Analog Input Module 464-8MA11, -8MF11, -8MB11 (Bipolar Fixed-Point Number)**

| Units | Measured Value in mV | | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| >4095 | 100.0 | 1000.0 | 2000.0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | Overflow |
| 4095 | 99.976 | 999.75 | 1999.5 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | Overrange |
| 2049 | 50.024 | 500.24 | 1000.48 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 2048 | 50.0 | 500.0 | 1000.0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Nominal range |
| 1024 | 25.0 | 250.0 | 500.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0.024 | 0.24 | 0.48 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 0 | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -1 | - 0.024 | -0.24 | -0.48 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| -1024 | - 25.0 | - 250.0 | - 500.0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -2048 | -50.0 | -500.0 | -1000.0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -2049 | -50.024 | -500.24 | -1000.48 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | Overrange |
| -4095 | -99.976 | -999.75 | -1999.5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| <-4095 | -100.0 | -1000.0 | -2000.0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Overflow |

464-8MA11/-8MA21 “No linearization” (4x±50 mV)
464-8MF11 (2x±500 mV)
464-8MB11 (4x±1 V)

**Table 11-6. Analog Input Module 464-8MC11, -8MD11 (Bipolar Fixed-Point Number)**

| Units | Measured Value in V | Measured Value in mA | High Byte | Low Byte | Range |
|---|---|---|---|---|---|
| >4095 | 20.000 | 40.0 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 1 | Overflow |
| 4095 | 19.995 | 39.9902 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | Overrange |
| 2049 | 10.0048 | 20.0098 | 0 1 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| 2048 | 10.000 | 20.0 | 0 1 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| 1024 | 5.000 | 10.0 | 0 0 1 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| 1 | 0.0048 | 0.0098 | 0 0 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| 0 | 0.0 | 0.0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | Nominal range |
| -1 | -0.0048 | -0.0098 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | |
| -1024 | - 5.000 | -10.0 | 1 1 1 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| -2048 | -10.000 | -20.0 | 1 1 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| -2049 | -10.0048 | -20.0098 | 1 0 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | Overrange |
| -4095 | -19.995 | - 39.9902 | 1 0 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| <-4095 | -20.000 | -40.0 | 1 0 0 0 0 0 0 0 | 0 0 0 0 1 0 0 1 | Overflow |

464-8MC11 (4x±10 V)
464-8MD11 (4x±20 mA)

**Table 11-7. Analog Input Module 464-8ME11, 4x4 to 20 mA (Absolute Value Representation)**

| Units | Measured Value in mA | High Byte | Low Byte | Range* |
|---|---|---|---|---|
| >4095 | > 32.769 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 1 | Overflow |
| 4095 | 31.992 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | Overrange |
| 2561 | 20.008 | 0 1 0 1 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| 2560 | 20.0 | 0 1 0 1 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| 2048 | 16.0 | 0 1 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | Nominal range |
| 512 | 4.0 | 0 0 0 1 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| 511 | 3.992 | 0 0 0 0 1 1 1 1 | 1 1 1 1 1 0 0 0 | |
| 384 | 3.0 | 0 0 0 0 1 1 0 0 | 0 0 0 0 0 0 0 0 | |
| 0 | 0.0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | Transducer |
| -1 | -0.008 | 1 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | failure? |
| <-4095 | <-32.769 | 1 0 0 0 0 0 0 0 | 0 0 0 0 1 0 0 1 | |

* Because of tolerances of components used in the module, the converted value can also be negative (e.g. $FFF8_H$ Unit: -1).

**Table 11-8. Analog Input Module 464-8MF11, 2x PT 100 (Unipolar)**
**Analog Input Module 464-8MF21, 2x PT 100 "No Linearization" (Unipolar)**

| Units | Resistance in | High Byte | Low Byte | Range |
|---|---|---|---|---|
| >4095 | 400.0 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 1 | Overflow |
| 4095 | 399.90 | 0 1 1 1 1 1 1 1 | 1 1 1 1 1 0 0 0 | Overrange |
| 2049 | 200.098 | 0 1 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| 2048 | 200.0 | 0 1 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |
| 1024 | 100.0 | 0 0 1 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | Nominal range |
| 1 | 0.098 | 0 0 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 | |
| 0 | 0.0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | |

**Table 11-9. Analog Input Module 464-8MF21, 2x PT 100 “with Linearization” (Bipolar)**

| Units | Resis-tance in | Temperature in | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ° C | ° F | | | | | | | | | | | | | | | | | |
| >1766 | >400 | >883 | >1531 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Overflow |
| 1766 | | 883 | 1531 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Overrange* |
| 1702 | | 851 | 1564 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | |
| 1700 | 390.26 | 850 | 1562 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 1400 | 345.13 | 700 | 1292 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1000 | 280.90 | 500 | 932 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 600 | 212.02 | 300 | 572 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 300 | 157.31 | 150 | 302 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 200 | 138.50 | 100 | 212 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Nominal range |
| 2 | 100.39 | 1 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 0 | 100.00 | 0 | 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -40 | 92.16 | -20 | -4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -80 | 84.27 | -40 | -40 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -200 | 60.25 | -100 | -148 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -202 | | -101 | -150 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | Overrange* |
| -494 | | -247 | -413 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | |
| <-494 | | <-247 | <-403 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Overflow |

* In the overrange area, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

**Table 11-10. Analog Input Module 464-8MA21, 4x±50 mV "with Linearization" and "with Temperature Compensation" (Bipolar); Thermoelement Type K (Nickel-Chromium/Nickel-Aluminium, according to IEC 584)**

| Units | Thermal Voltage in mV* | Temperature | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ° C | ° F | | | | | | | | | | | | | | | | | |
| >2359 | | | | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | Overflow |
| 1370 | | 1370 | 2498 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | Overrange** |
| 1369 | 54.773 | 1369 | 2496 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 1000 | 41.269 | 1000 | 1832 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 500 | 20.640 | 500 | 932 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 150 | 6.137 | 150 | 302 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | |
| 100 | 4.095 | 100 | 212 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Nominal range |
| 1 | 0.039 | 1 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -1 | -0.039 | 1 | -30 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| -100 | -3.553 | -100 | -148 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| -101 | -3.584 | -101 | -150 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Accuracy |
| -150 | -4.912 | -150 | -238 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 2 K |
| -200 | -5.891 | -200 | -328 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -201 | | -201 | -330 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | Overrange** |
| -273 | | | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | Overflow |
| X | | X | | X | X | X | X | X | X | X | X | X | X | X | X | X | 0 | 1 | 0 | Wire break |

This value corresponds to the terminal temperature at wire break

* For a reference temperature of 0° C (32° F)

** In the overrange area, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

**Table 11-11. Analog Input Module 464-8MA21, 4x±50 mV "with Linearization" and "with Temperature Compensation" (Bipolar); Thermoelement Type J (Iron/Copper-Nickel (Konstantan), according to IEC 584)**

| Units | Thermal Voltage in mV* | Temperature | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ° C | ° F | | | | | | | | | | | | | | | | | |
| 1485 | | | | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | Overflow |
| 1201 | | 1201 | 2194 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Overrange** |
| 1200 | 69.536 | 1200 | 2192 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1000 | 57.942 | 1000 | 1832 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 500 | 27.388 | 500 | 932 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 100 | 5.268 | 100 | 212 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0.05 | 1 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | Nominal range |
| 0 | 0 | 0 | 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -1 | -0.05 | -1 | -30 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| -100 | -4.632 | -100 | -148 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| -150 | -6.499 | -199 | -238 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| -199 | -7.868 | -200 | -326 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | |
| -200 | -7.890 | -200 | -328 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -201 | | -201 | -330 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | Overrange** |
| -273 | | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | Overflow |
| X | | | X | X | X | X | X | X | X | X | X | X | X | X | X | X | 0 | F | 0 | Wire break |

This value corresponds to the terminal temperature at wire break

* For a reference temperature of 0° C (32° F)

** In the overrange area, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

**Table 11-12. Analog Input Module 464-8MA21, 4x±50 mV "with Linearization" and "with Temperature Compensation" (Bipolar); Thermoelement Type L (Iron/Copper-Nickel (Konstantan), according to DIN 43710)**

| Units | Thermal Voltage in mV* | Temperature | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ° C | ° F | | | | | | | | | | | | | | | | | |
| 1361 | | | | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | Overflow |
| 901 | | 901 | 1654 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | Overrange** |
| 900 | 53.14 | 900 | 1652 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 500 | 27.85 | 500 | 932 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 250 | 13.75 | 250 | 482 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| 100 | +5.37 | 100 | 212 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Nominal range |
| 1 | 0.05 | 1 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 32 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| -1 | -0.05 | -1 | -30 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | |
| -100 | -4.75 | -100 | -148 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| -150 | -6.60 | -150 | -238 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | |
| -190 | -7.86 | -190 | -310 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| -199 | -8.12 | -199 | -326 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | |
| -200 | | -200 | -328 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | Overrange** |
| -273 | | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | Overflow |
| X | | X | | X | X | X | X | X | X | X | X | X | X | X | X | X | 0 | 1 | 0 | Wire break |

This value corresponds to the terminal temperature at wire

* For a reference temperature of 0° C (32° F)

** In the overrange area, the current slope of the characteristic curve is maintained when leaving the linearized nominal range.

**Analog value representation of analog input module 466-8MC11**

The **466-8MC11** analog input module stores each analog value in a single byte. The other analog input modules store the analog values in words (see Table 11-4).

**Table 11-13. Analog Input Module 466-8MC11, 4x 0 to 10 V**

| Units | Voltage in mV | Bit Representation |
|---|---|---|
| 255 | 9961 | 1 1 1 1 1 1 1 1 |
| 254 | 9922 | 1 1 1 1 1 1 1 0 |
| . | . | . |
| 128 | 5000 | 1 0 0 0 0 0 0 0 |
| . | . | . |
| 1 | 39 | 0 0 0 0 0 0 0 1 |
| 0 | 0 | 0 0 0 0 0 0 0 0 |

If you want to read in the analog value with function block FB250 (analog value reading), you have to pre-process the analog value before calling up FB250.

**Example 1:**

Analog input module 466-8MC11 is inserted in slot 1, which means that the module's start address is 72.

The analog values are stored in four consecutive bytes:

| | | |
|---|---|---|
| 1st analog value | (channel 0) | in IB72 |
| 2nd analog value | (channel 1) | in IB73 |
| 3rd analog value | (channel 2) | in IB74 |
| 4th analog value | (channel 3) | in IB75 |

Function block FB72, pictured below, reads in the analog values and pre-processes them for function block FB250 (analog value reading).

| FB72 | Explanation |
|---|---|
| NAME :READ 466 | |
| | READ IN ALL CHANNELS |
| 0005 : | OF AI 466 |
| 0006 :L IW 72 | READ ALL FOUR CHANNELS |
| 0007 :T FW 72 | AND REARRANGE |
| 0008 :L IW 74 | |
| 0009 :T FW 74 | |
| 000A : | |
| 000B :L FY 72 | PROCESS EACH ANALOG VALUE |
| 000C :SLW 6 | AND REWRITE THEM IN |
| 000D :T IW 72 | THE PII SO THAT FB250 |
| 000E : | CAN ACCESS THEM |
| 000F :L FY 73 | WITHIN THAT SCAN. |
| 0010 :SLW 6 | |
| 0011 :T IW 74 | |
| 0012 : | |
| 0013 :L FY 74 | |
| 0014 :SLW 6 | |
| 0015 :T IW 76 | |
| 0016 : | |
| 0017 :L FY 75 | |
| 0018 :SLW 6 | |
| 0019 :T IW 78 | |
| 001A : | |
| 001B :BE | |

**Example 2:**

Analog input module 466-8MC11 is inserted in slot 0, which means that the module's start address is 64.

The analog values that are read in are stored in four consecutive bytes:

| | | |
|---|---|---|
| 1st analog value | (channel 0) | in IB64 |
| 2nd analog value | (channel 1) | in IB65 |
| 3rd analog value | (channel 2) | in IB66 |
| 4th analog value | (channel 3) | in IB67 |

Function block 73, pictured below, reads in the analog values and pre-processes them for FB250. The additional processing with FB250 is done just like module 464, however without an overflow bit.

| FB73 | Explanation |
|---|---|
| NAME :READ AI | |
| : | |
| : | |
| 000A :L IB 67 | Read in channel 3 |
| 000C :SLW 6 | |
| 000E :T IW 70 | |
| 0010 : | |
| 0012 :L IB 66 | Read in channel 2 |
| 0014 :SLW 6 | |
| 0016 :T IW 68 | |
| 0018 : | |
| 001A :L IB 65 | Read in channel 1 |
| 001C :SLW 6 | |
| 001E :T IW 66 | |
| 0020 : | |
| 0022 :L IB 64 | Read in channel 0 |
| 0024 :SLW 6 | |
| 0026 :T IW 64 | |
| 0028 : | |
| 002A : | |
| 002C :JU FB 250 | |
| 002E NAME :RLG:AI | |
| 0030 BG : KF +0 | Module on slot 0 |
| 0032 KNKT : KY 0,4 | Channel/No. 0, unipolar representation |
| 0034 OGR : KF +1000 | Upper limit 1000 (1000 mV) |
| 0036 UGR : KF +0 | Lower limit 0 |
| 0038 EINZ : F 0.0 | No meaning |
| 003A XA : FW 100 | Output, 0 to 1000 mV in KF |
| 003C FB : F 102.0 | Error bit for parameter assignment |
| 003E BU : F 102.1 | Range overflow |
| 0040 : | (always 0 with this module) |
| 0042 :BE | |

## 11.5 Analog Output Modules

Analog output modules convert the bit patterns that are output by the CPU into analog output voltages or currents.

### 11.5.1 Connection of Loads to Analog Output Modules

No adjustments are necessary if you want to connect loads to the analog outputs.

Check the following items before connecting loads:

- The load voltage 24 V DC must be connected to terminals 1 and 2.
- The maximum permissible potential difference between the outputs is 60 V AC.
- Unused outputs must be left open-circuited.

Figure 11-9 shows how to connect loads to the voltage outputs of the following modules:

- 470-8MA12 (2x±10 V)
- 470-8MD12 (2x+1 to 5 V)

The sensor lines (S+ and S-) must be directly connected to the load, so that the voltage is measured and regulated directly at the load. In this manner, voltage drops of up to 3 V per line can be compensated for.

The sensor lines can be left out if the resistances of the QV and M lines are negligible compared to the load resistance. In such a case, connect terminal S+to terminal QV, and terminal S- to $M_{ANA}$.

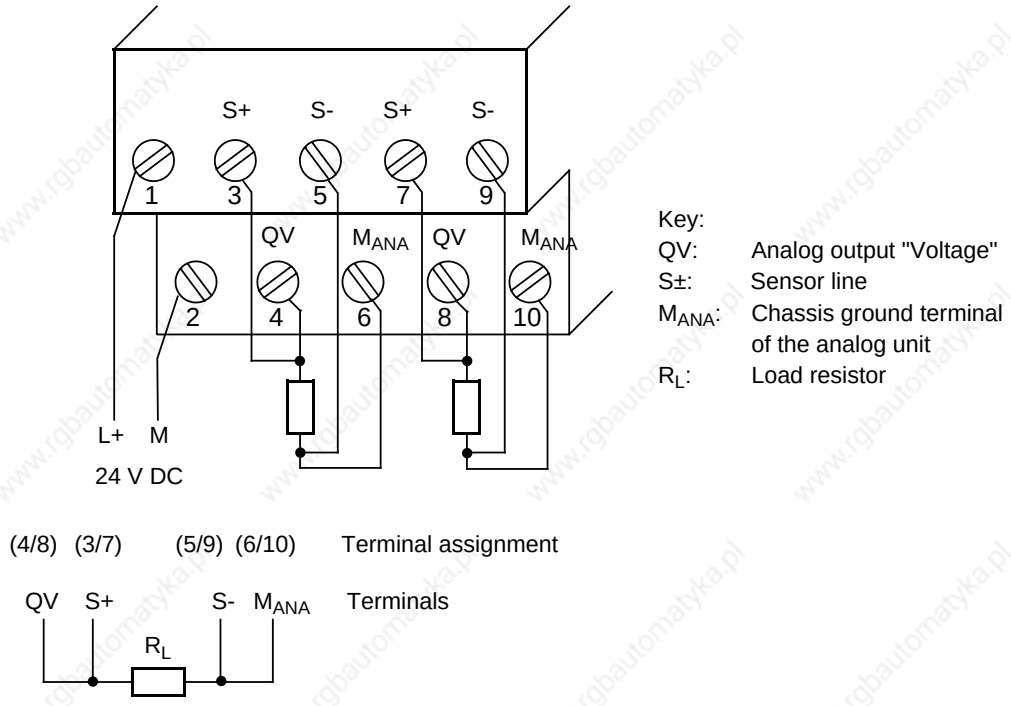



**Figure 11-9. Load Connection via a Four-Wire Circuit (6ES5 470-8MA12 or 6ES5 470-8MD12)**

Figure 11-10 shows how to connect loads to the current outputs of the following modules.

- 470-8MB12 (2x±20 mA)
- 470-8MC12 (2x+4 to 20 mA).

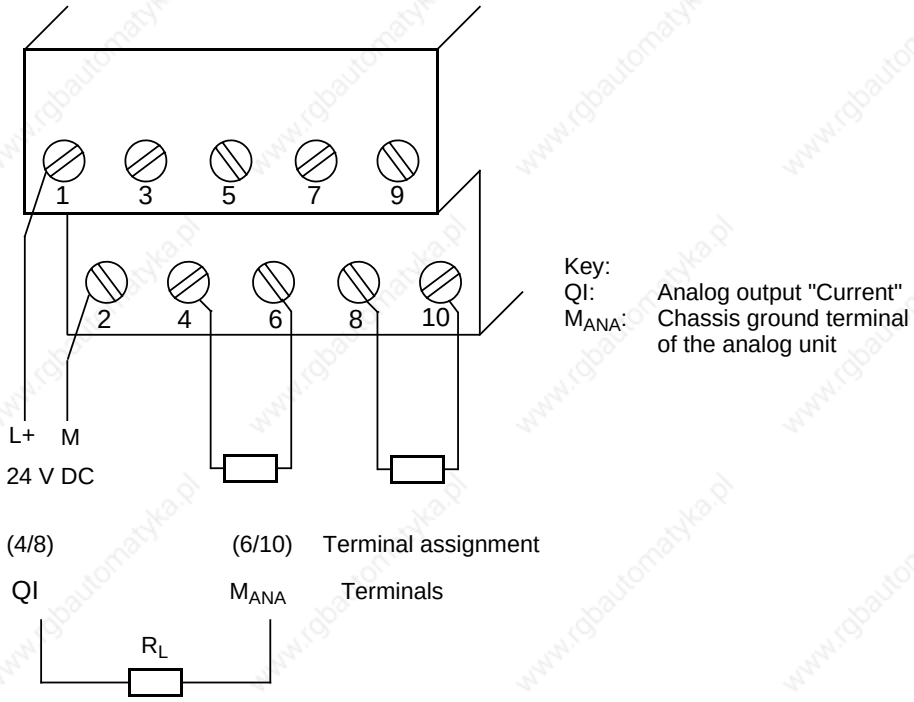



**Figure 11-10. Load Connection via a Two-Wire Circuit (6ES5 470-8MB12 or 6ES5 470-8MC12)**

## 11.5.2 Analog Value Representation of Analog Output Modules

Table 11-14 shows how the analog output value has to be stored in the process image output table (PIQ).

**Table 11-14. Representation of an Analog Output Value as a Bit Pattern**

| | High Byte | | | | | | | | Low Byte | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Analog value represent. | S | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ | X | X | X | X |

Key: S sign bit
X irrelevant bits

Table 11-15 and 11-16 show the voltage and currents assigned to the bit patterns.

**Table 11-15. Output Voltages and Currents for Analog Output Modules (Fixed-Point Number Bipolar)**

| Units | Output Values | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | in V | in mA | | | | | | | | | | | | | | | | | |
| 1280 | 12.5 | 25.0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | Overrange |
| 1025 | 10.0098 | 20.0195 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| 1024 | 10.0 | 20.0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| 512 | 5.0 | 10.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| 1 | 0.0098 | 0.0195 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| 0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -1 | -0.0098 | -0.0195 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | x | x | x | x | Nominal range |
| -512 | -5.0 | -10.0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -1024 | -10.0 | -20.0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -1025 | -10.0098 | -20.0195 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | x | x | x | x | |
| -1280 | -12.5 | -25.0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | Overrange |

2x±10 V 6ES5 470-8MA12
2x±20 mA 6ES5 470-8MB12

**Table 11-16. Output Voltages and Currents for Analog Output Modules (Unipolar)**

| Units | Output Values | | High Byte | | | | | | | | Low Byte | | | | | | | | Range |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | in V | in mA | | | | | | | | | | | | | | | | | |
| 1280 | 6.0 | 24.0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | Overflow |
| 1025 | 5.004 | 20.016 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | |
| 1024 | 5.0 | 20.0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| 512 | 3.0 | 12.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | Nominal range |
| 1 | 1.004 | 4.016 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | x | x | x | x | |
| 0 | 1.0 | 4.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -1 | 0.996 | 3.984 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | x | x | x | x | |
| -256 | 0.0 | 0.0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -512 | -1.0 | -4.0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | Overrange |
| -1024 | -3.0 | -12.0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |
| -1280 | -4.0 | -16.0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | x | x | x | |

2x 1 to 5 V 6ES5 470-8MD12
2x 4 to 20 mA 6ES5 470-8MC12

## 11.6 Analog Value Conversion: Function Blocks FB250 and FB251

### 11.6.1 Reading in and Scaling an Analog Value - FB250 -

Function block FB250 reads in an analog value from an analog input module and outputs a value XA in the scale range specified by the user.

Specify the type of analog value representation for the module (channel type) in the KNKT parameter (see Table 11-17). Define the desired range using the “upper limit” (OGR) and "lower limit” (UGR) parameters.

**Table 11-17. Call and Parameter Assignments of FB250**

| Parameter | Explanation | Type | Assignment | STL |
|---|---|---|---|---|
| BG | Slot number | D KF | 0 to 7 | : JU FB 250<br>NAME : RLG:AI<br>BG :<br>KNKT :<br>OGR :<br>UGR :<br>EINZ :<br>XA :<br>FB :<br>BU : |
| KNKT | Channel number<br>Channel type | D KY | KY = x,y<br>x = 0 to 3<br>y = 3 to 6<br>3: Absolute value representation (4 to 20 mA)<br>4: Unipolar representation<br>5: Bipolar absolute value<br>6: Bipolar fixed-point number | |
| OGR | Upper limit of the output value | D KF | -32767 to +32767 | |
| UGR | Lower limit of the output value | D KF | -32767 to +32767 | |
| EINZ | Single scan | I BI | Not relevant | |
| XA | Output value | Q W | Scaled analog value is “0” on wirebreak | |
| FB | Error bit | Q BI | “1” on wirebreak, illegal channel or slot number or illegal channel type | |
| BU | Range violation | Q BI | “1” when nominal range is exceeded | |

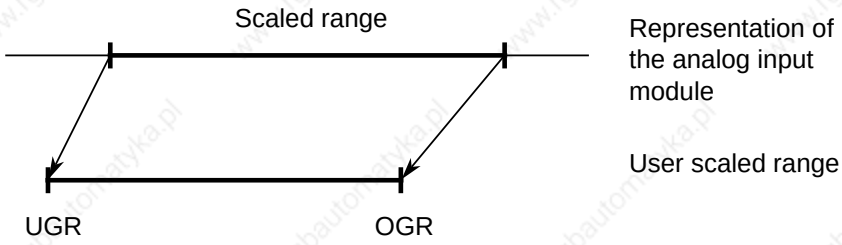



**Figure 11-11. Scaling Schematic for FB250**

**Example:** Display of Tank Make-Up Quantity

The make-up of a cylindrical tank holding 30 $m^3$ is to be shown on a 3-digit display. The individual digits must be set in BCD.

The level of the liquid in the tank is sensed by a SONAR-BERO®, range 80 to 600 cm, with analog output (see Catalog NS3).

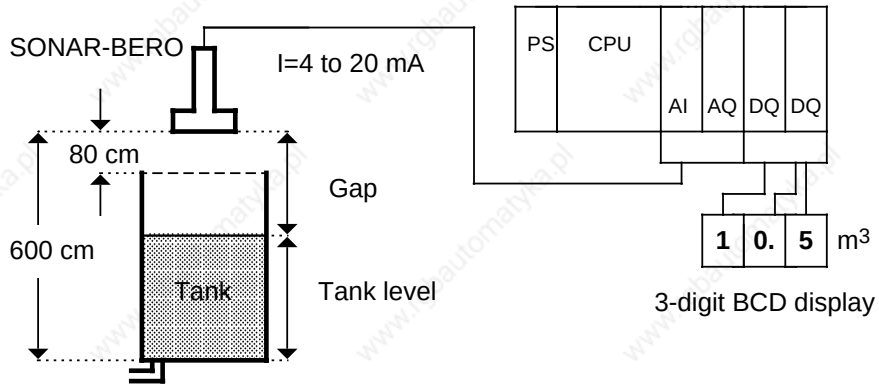



**Figure 11.12 Schematic for "Display of Tank Make-Up Quantity"**

The analog output of the SONAR-BERO delivers a constant current in the range 4 to 20 mA proportional to the gap between sensor and liquid. This current is routed to the 4 to 20 mA analog input module in slot 0, channel 0.

FB250 converts the range 4 to 20mA to the range 0 to 30.0 $m^3$.
The value is stored in flag word 1 as a fixed-point number. Initialization takes place in the calling block. FB241 converts the fixed-point number into a BCD number.

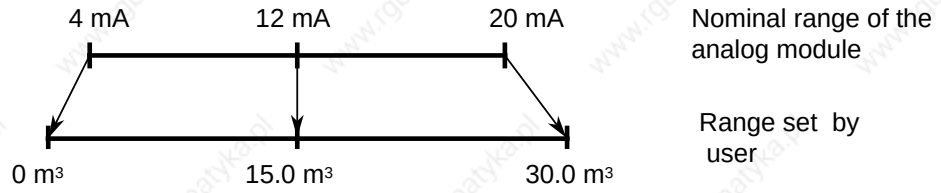



**Figure 11-13. Conversion of the Nominal Range into the Defined Range**

| STL | Explanation |
|---|---|
| JU FB 250 | Unconditional call FB250 |
| NAME : RLG:AI | |
| BG : 0 | Slot 0 |
| KNKT : 0.3 | Channel 0, channel type 3 |
| OGR : 300 | Upper limit: 30.0 $m^3$ |
| UGR : 0 | Lower limit: 0.0 $m^3$ |
| EINZ : | No meaning |
| XA : FW1 | Make-up quantity stored in flag word 1 as fixed-point number |
| FB :F0.0 | "1", if wire break |
| BU :F0.1 | "1", if tank too full |
| JU FB 241 | Conversion of fixed-point number into |
| . . . | BCD number |

The BCD number is stored in flag bytes 11 to 13. Output is via two 8-channel digital output modules in slots 2 and 3. The BCD tetrads 5 and 6 stored in flag word 11 need not be output since the number has only three digits.

| STL | Explanation |
|---|---|
| . . . | |
| L FW12 | Read tetrads 0 to 3 of the BCD number |
| T QW2 | and transfer to output modules. |
| BE | |

## 11.6.2 Output of Analog Value - FB251 -

Analog values can be output to analog output modules using this function block. In doing so, values from the range between the lower limit (UGR) and high limit (OGR) parameters are converted to the nominal range of the module in question.

**Table 11-18. Call and Parameter Assignments of FB251**

| Parameter | Explanation | Type | Assignment |
|---|---|---|---|
| XE | Analog value to be output | I W | Input value (two's complement) in the range UGR to OGR |
| BG | Slot address | D KF | 0 to 7 |
| KNKT | Channel number<br>Channel type | D KY | KY = x,y<br>x = 0;1<br>y = 0;1<br>0: unipolar repre-sentation<br>1: bipolar fixed-point number |
| OGR | Upper limit of the output value | D KF | -32767 to +32767 |
| UGR | Lower limit of the output value | D KF | -32767 to +32767 |
| FEH | Error in limit value setting | Q BI | "1" if UGR = OGR, invalid channel or slot, or invalid channel type |
| BU | Input value exceeds UGR or OGR | Q BI | "1" if XE lies outside limits (UGR; OGR). XE assumes the limit value |

```
        : JU FB 251
NAME    : RLG:AQ
XE      :
BG      :
KNKT    :
OGR     :
UGR     :
FEH     :
BU      :
```

**Example:** Display of Tank Contents on an Analog Measuring Instrument

The make-up quantity of a 30 $m^3$ tank is stored in flag word 1 as a fixed-point number (see example FB250). The ± 20 mA analog output module in slot 1, channel 0, transfers the standardized value to the measuring instrument. The value is displayed within the range 0 to 20 mA.

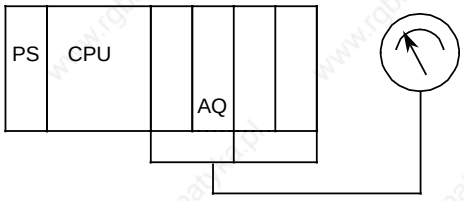



**Figure 11-14. Schematic for "Display of Tank Contents"**

The tank contents are determined from the make-up quantity.

| STL | Explanation |
|---|---|
| `L KF +300`<br>`L FW 1`<br>`-F`<br>`T FW 20`<br>`...` | Maximum tank capacity<br>Make-up quantity<br>Calculate difference<br>Store tank contents in FW20 |

The UGR and OGR parameters of FB 251 refer to the nominal range of the analog output module. For this reason, the UGR parameter must be assigned the value -30.0.

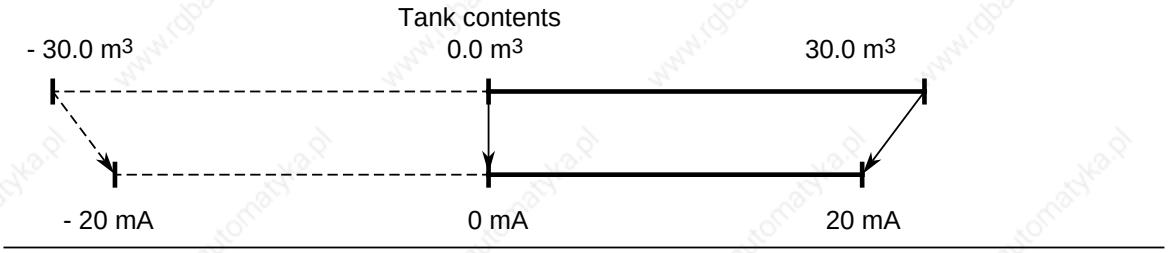



**Figure 11-15. Transformation of the Analog Value to the Nominal Range**

| STL | Explanation |
|---|---|
| `...`<br>`JU FB251`<br>`NAME :RLG:AQ`<br>`XE :FW20`<br>`BG :1`<br>`KNKT :0.1`<br>`OGR :300`<br>`UGR :-300`<br>`FEH :F0.2`<br>`BU :F0.3`<br>`BE` | Unconditional call-up FB251<br><br>Tank contents<br>Slot 1<br>Channel 0, channel type 1<br>High limit 30.0 m3<br>Low limit - 30.0 m3<br>“1”, if wire break<br>“1”, if tank too full |

## 12 The Integral Real-Time Clock, for CPU 103 Version 8MA02 and Higher


12.1 Function . . . . . 12 - 1

12.2 Setting Parameters in DB1, for CPU 103 Version 8MA03 and Higher . . . . . 12 - 2
12.2.1 Defaults . . . . . 12 - 2
12.2.2 Reading the Current Clock Time and the Current Date . . . . . 12 - 3
12.2.3 DB1 Parameters Used for the Integral Real-Time Clock . . . . . 12 - 4

12.3 Programming the Integral Real-Time Clock in DB1, for CPU 103 Version 8MA03 and Higher . . . . . 12 - 5
12.3.1 Setting the Clock in DB1 . . . . . 12 - 5
12.3.2 Setting the Prompt Time in DB1 . . . . . 12 - 6
12.3.3 Setting the Operating Hours Counter in DB1 . . . . . 12 - 7
12.3.4 Entering the Clock Time Correction Factor in DB1 . . . . . 12 - 7

12.4 Structure of the Clock Data Area . . . . . 12 - 8

12.5 Structure of the Status Word and How to Scan it . . . . . 12 - 12

12.6 Setting Parameters for the Clock Data Area and the Status Word in the System Data Area . . . . . 12 - 15

12.7 Programming the Integral Real-Time Clock in the User Program . . . . . 12 - 21
12.7.1 Reading and Setting the Clock . . . . . 12 - 21
12.7.2 Programming the Prompt Function . . . . . 12 - 25
12.7.3 Programming the Operating Hours Counter . . . . . 12 - 30
12.7.4 Entering the Clock Time Correction Factor . . . . . 12 - 35

## Figures


12-1 DB1 with Default Parameters for Integral Real-Time Clock . . . . . 12 - 2
12-2 Example: Setting the Clock in DB1 to Monday, November 9, 1992, 15:30 . . . 12 - 5
12-3 Example: Setting the Prompt Time in DB1 to Thursday, December 17, 1992, 8:00 o'clock . . . . . 12 - 6
12-4 Setting the Operating Hours Counter in DB1 to 1600 Hours . . . . . 12 - 7
12-5 Entering a Correction Factor of +90s in DB1 . . . . . 12 - 7
12-6 How DB1 or the Control Program and the Clock Access the Clock Data Area . . . . . 12 - 8
12-7 Flowchart - Transferring Time and Date Settings to the Clock . . . . . 12 - 20
12-8 Flowchart - Transferring a New Prompt Time . . . . . 12 - 26
12-9 Flowchart - Transferring Settings to the Operating Hours Counter . . . . . 12 - 31


## Tables


12-1 Reading the Current Clock Time and Current Date . . . . . 12 - 3
12-2 DB1 Parameters for the Integral Real-Time Clock . . . . . 12 - 4
12-3 Clock Data in the Clock Data Area . . . . . 12 - 9
12-4 Range Definitions for Clock Data . . . . . 12 - 10
12-5 Significance of Bits 0, 1, 2, and 3 of the Status Word . . . . . 12 - 13
12-6 Significance of Bits 4 and 5 of the Status Word . . . . . 12 - 13
12-7 Significance of the Operating Hours Counter Flags Bits 8, 9, and 10 of the Status Word . . . . . 12 - 14
12-8 Significance of the Prompting Time Flags Bits 12, 13, and 14 of the Status Word . . . . . 12 - 14
12-9 The System Data Area for the Integral Real-Time Clock . . . . . 12 - 15
12-10 FB1 Program . . . . . 12 - 17
12-11 OB21 Program . . . . . 12 - 18
12-12 OB22 Program . . . . . 12 - 18
12-13 DB75 Program . . . . . 12 - 18
12-14 FORCE VAR Function . . . . . 12 - 19

# 12 The Integral Real-Time Clock, for CPU 103 Version 8MA02 and Higher

## 12.1 Function

The integral real-time clock offers the following possibilities of controlling the process sequence:

- Clock and calendar function
  - Used to configure clock-time dependent control, for example
- Prompt and alarm function
  - Used to monitor the duration of a process, for example
- Operating hours counter
  - Used to monitor inspection intervals, for example

The clock begins running when you supply voltage to the programmable controller. The default is April 1, 1992, 12:00 o'clock. You set the clock by setting its parameters.

There are two possibilities.

- With CPU 103 version 8MA03, you can set the clock parameters in DB1 (see section 12.2).
- With CPU 103 version 8MA02 and higher, you can set the clock parameters in the system data area (see section 12.6) and program the clock in the user program (see section 12.7).

The hardware clock requires a clock data area and a status word in order to function. The location of both the clock data area and the status word must be stored in system data 8 to 10.

**Operating Principle of the Clock**

Data exchange between the integral real-time clock and the control program always goes through the clock data area. The clock stores current values for time, date, and operating hours counter in the clock data area. You can transfer into the clock data area the values for the time, date, prompt time, and operating hours counter that you want the clock to use.

You can scan the status word to identify setting errors, for example. Or you can change certain status word bits to deliberately disable or enable transfer or read operations.

Refer to sections 12.4 and 12.5 for additional information about the clock data area and the status word. These sections are especially important if you want to set clock parameters in the system data. If you are new to SIMATIC, you may prefer to set clock parameters in DB1.

## 12.2 Setting Parameters in DB1, for CPU 103 Version 8MA03 and Higher

Set the clock parameters in DB1 to be able to use the clock functions. Follow the same rules you used in setting parameters for other functions. Refer to section 9.1.

**Procedures for Setting Parameters in DB1**

1. Perform an overall reset.
2. Output default DB1 to the programmer.
3. Use the cursor to jump into the clock parameter block.
4. Change the parameters.
5. Transfer the changed DB1 to the programmable controller.
6. Switch the programmable controller from STOP to RUN.

Every time there is a change from STOP to RUN, the programmable controller accepts the new clock data.

> **Note**
>
> The system data contents are deleted during an overall reset. The clock continues to run internally with the current values.
> The clock time is updated one second after the next cycle starts.

### 12.2.1 Defaults

The following values are preset in the parameter block when you output the default DB1.

```
 36:     KS  ='PGN 01 ; #CLP: CF 0     ';
 48:     KS  ='CLK DB5  DWO   STW      ';
 60:     KS  ='MW102       STP Y SAV Y ';
 72:     KS  ='OHE N    SET 4 01.04.92 ';
 84:     KS ='12:00:00     TIS 4       ';
 96:     KS  ='01.04. 13:00:00     OHS ';
108:     KS  ='000000:00:00 # ; SDP: WD';
```

**Figure 12-1. DB1 with Default Parameters for Integral Real-Time Clock**

After the CLP block ID for the integral real-time clock, the CLK parameter defines the location of the clock data (in DB5 beginning with DW0, for example). The STW parameter specifies the location of the status word (in flag word MW102, for example). You must specify both parameters if you want to read the clock.

Section 12.2.2 describes the procedures you must follow to read the clock.

Section 12.2.3 lists all of the parameters that you can use for the integral clock.

### 12.2.2 Reading the Current Clock Time and the Current Date

Proceed as follows to see how and with which values the clock runs.

1. Perform an overall reset.
2. Output DB1 to the programmer.
3. Overwrite both (#) comment characters with a blank space.
4. Generate DB5 with DW0 to DW21. See Table 12-3 for information about storing the current clock time and current date.
5. Switch the programmable controller from STOP to RUN. The clock accepts the values present in DB1.
6. Enter DB5 and DW0 to DW3 on the programmer by using the FORCE VAR function.
7. Press the “ENTER” key twice. The clock runs using the current values.

**Table 12-1. Reading the Current Clock Time and Current Date**

| Operand | Signal States | Explanation |
|---|---|---|
| DB 5 | | |
| DW 0 | KH = 0004 | Wednesday |
| DW 1 | KH = 0104 | October 1 |
| DW 2 | KH = 9212 | 1992, 12:00 |
| DW 3 | KH = 0000 | |

### 12.2.3 DB1 Parameters Used for the Integral Real-Time Clock

**Table 12-2. DB1 Parameters for the Integral Real-Time Clock**

| Parameters | Argument | Meaning |
|---|---|---|
| **Block ID: CLP:** | | **Clock Parameters** |
| CF | p | Entering the correction factor (***C***orrection ***F***actor) |
| CLK | DBxDWy, MWz,EWv or AWv | Location of the clock data (***CL***oc***K*** Data) |
| STW | DBxDWy, MYz,EWv or AWv | Location of the status word (***ST***atus ***W***ord) |
| STP | J/Y/N | Updating the clock during STOP (***ST***o***P)*** |
| SAV | J/Y/N | Saving the clock time after the last change from RUN to STOP or Power OFF (***SAV***e) |
| OHE | J/Y/N | Enabling the operating hours counter (***O***perating ***H***ours counter ***E***nable) |
| SET | wd dd.mm.yy hh:mn:ss[1] AM/PM[2] | Setting the clock time and date |
| TIS | wd dd.mm. hh:mn:ss[1] AM/PM[2] | Setting the prompt time (***TI***mer ***S***et) |
| OHS | hhhhhh:mn:ss[1] | Setting the operating hours counter (***O***perating ***H***ours counter ***S***et) |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| wd | = | 1 | to | 7 | (weekday = Sun. - Sat.) | p | = | – 400 | to 400 |
| dd | = | 01 | to | 31 | (day) | v | = | 0 | to 126 |
| mm | = | 01 | to | 12 | (month) | x | = | 2 | to 255 |
| yy | = | 0 | to | 99 | (year) | y | = | 0 | to 255 |
| hh | = | 00 | to | 23 | (hours) | z | = | 0 | to 254 |
| mn | = | 00 | to | 59 | (minutes) | j/J | = | yes | |
| ss | = | 00 | to | 59 | (seconds) | y/Y | = | yes | |
| hhhhhh | = | 0 | to | 999999 | (hours) | n/N | = | no | |

[1] If an argument such as seconds, for example, is not to be entered, enter XX. The clock continues to run with the current data. The TIS parameter block does not acknowledge this argument.

[2] If you enter AM or PM after the clock time, the clock runs in the 12-hour mode. If you omit this argument, the clock runs in the 24-hour mode. You must use the same time mode in the SET and TIS parameter blocks.

## 12.3 Programming the Integral Real-Time Clock in DB1, for CPU 103 Version 8MA03 and Higher

Sections 12.3.1 to 12.3.4 contain examples for programming the clock in DB1. Adhere to the rules described in chapter 9 for setting parameters when you enter these examples into the programmable controller.

> **Note**
>
> If the programmable controller recognizes a parameter setting error in DB1, the programmable controller remains in the STOP mode even after it has been switched from STOP to RUN. The red LED is lit.

### 12.3.1 Setting the Clock in DB1

**How to set the clock in DB1**

1. Perform an overall reset on the programmable controller.
2. Generate DB5 with DW0 to DW21.
3. Output default DB1 to the programmer.
4. Overwrite the comment characters (#) with a blank space.
5. Use the cursor to jump to the CLP parameter block.
6. Enter the example.
   - Set the clock to the following date: Monday, November 9, 1992, 15:30.

| Setting the Clock | Explanation |
|---|---|
| 36: KS ='PGN 01 ; CLP: CF 0 ';<br>48: KS ='CLK DB5 DW0 STW ';<br><br><br>60: KS ='MW102 STP Y SAV Y ';<br><br><br>72: KS ='OHE N SET 2 09.11.92 ';<br><br><br>84: KS ='15:30:00 TIS 4 '; | The clock data is stored in data block 5 beginning with data word DW0. The status word is located in flag word MW 102.<br>The clock is updated when the programmable controller is in the STOP mode. The clock time is saved in the clock data area. See Table 12-3.<br>After the SET parameter, enter the weekday, the date, and the clock time you want the clock to use when it begins running. Be certain to include the blank spaces.<br>The clock runs in the 24-hour time mode since you do not enter either AM or PM. |

**Figure 12-2. Example: Setting the Clock in DB1 to Monday, November 9, 1992, 15:30**

7. Transfer the changed DB1 to the programmable controller.
8. Switch the programmable controller from STOP to RUN.

Each time the programmable controller is switched from STOP to RUN, it accepts the new clock data.

## 12.3.2 Setting the Prompt Time in DB1

**How to set the prompt time in DB1**

1. Perform an overall reset on the programmable controller.
2. Generate DB5 with DW0 to DW21.
3. Output default DB1 to the programmer.
4. Overwrite the comment characters (#) with a blank space.
5. Use the cursor to jump into the CLP parameter block.
6. Enter the example.
   - Set the clock to the following prompt time: Thursday, December 17, 1992, 8:00 o'clock.

| Setting the Prompting Time | Explanation |
|---|---|
| 36: KS ='PGN 01 ; CLP: CF 0 ';<br>48: KS ='CLK DB5 DW0 STW ';<br>60: KS ='MW102 STP Y SAV Y ';<br>84: KS ='12:00:00 TIS 5 ';<br>96: KS ='17.12 08:00:00 PM OHS '; | The clock data is stored in data block 5 beginning with data word DW0. The status word is located in flag word MW 102.<br>The clock is updated when the programmable controller is in the STOP mode. The clock time is saved in the clock data area. See Table 12-3.<br>After the parameter for TIS, enter the weekday, date and time to initiate the prompt time.<br>You can enter the parameter for the clock mode. The clock runs in the 24-hour time mode. |

**Figure 12-3. Example: Setting the Prompt Time in DB1 to Thursday, December 17, 1992, 8:00 o'clock**

7. Transfer the changed DB1 to the programmable controller.
8. Switch the programmable controller from STOP to RUN.

Each time the programmable controller is switched from STOP to RUN, it accepts the new clock data.

### 12.3.3 Setting the Operating Hours Counter in DB1

**How to set the operating hours counter in DB1**

1. Perform an overall reset on the programmable controller.
2. Generate DB5 with DW0 to DW21.
3. Output default DB1 to the programmer.
4. Overwrite the comment characters (#) with a blank space.
5. Use the cursor to jump to the CLP parameter block.
6. Enter the example.
   - You are setting the start value for the operation hours counter to 1600 hours.

| Setting the Operating Hours Counter | Explanation |
|---|---|
| 36: KS ='PGN 01 ; CLP: CF 0 ';<br>48: KS ='CLK DB5 DW0 STW ';<br><br>60: KS ='MW102 STP Y SAV Y';<br><br><br>72: KS ='OHE Y SET 4 01.04.92 ';<br>.<br>.<br>.<br>96: KS ='01.04. 13:00:00 OHS ';<br>108: KS ='001600:00:00 ; SDP: WD'; | The clock data is stored in data block 5 beginning with data word DW0. The status word is located in flag word MW 102.<br>The clock is updated when the programmable controller is in the STOP mode. The clock time is saved in the clock data area. See Table 12-3.<br>The operating hours counter is enabled.<br><br><br>After the OHS parameter, enter the start value for the operating hours counter. |

**Figure 12-4. Setting the Operating Hours Counter in DB1 to 1600 Hours**

7. Transfer the changed DB1 to the programmable controller.
8. Switch the programmable controller from STOP to RUN.

Each time the programmable controller is changed from STOP to RUN, it accepts the new clock data.

### 12.3.4 Entering the Clock Time Correction Factor in DB1

The exactness of the clock is temperature-dependent. You can configure a correction value to increase the clock's exactness. The correction value is output in s/month. You must measure how many seconds per month the clock runs fast or slow. A month is defined as 30 days.

**Example:** Your measurements indicate the clock is 12 s too slow in 4 days. That would be 90 s too slow in 30 days. The correction value is +90 s/month.

In addition to the changed clock parameters, enter the example into DB1 as follows:

| Enter the Clock Time Correction Factor | Explanation |
|---|---|
| 36: KS ='PGN 01 ; CLP: CF +90 ' ; | The correction value of +90 s is loaded into the clock. |

**Figure 12-5. Entering a Correction Factor of +90 s in DB1**

## 12.4 Structure of the Clock Data Area

You need only to change the default values in DB1 to program the clock in DB1. See section 12.2. During start-up, the DB1 interpreter writes all information into the system data area.

**TIP**: Do not attempt to set parameters in the system data, or to access directly from the user program unless you have extensive knowledge of the system.

You must store the location of the clock data area in system data 8 and 9.

Data exchange between DB1 or the control program and the integral real-time clock is always through the clock data area.

- The integral real-time clock stores current time, date, and operating hours counter values in the clock data area (flag area, data block, input area, or output area).
- DB1 and the control program store the settings for prompt times and operating hours counters in the same data area.

The control program can only read from or write to the clock data area. The control program can never access the clock directly. Figure 12-6 illustrates the relationship between DB1 or the control program, the clock data area, and the integral real-time clock.

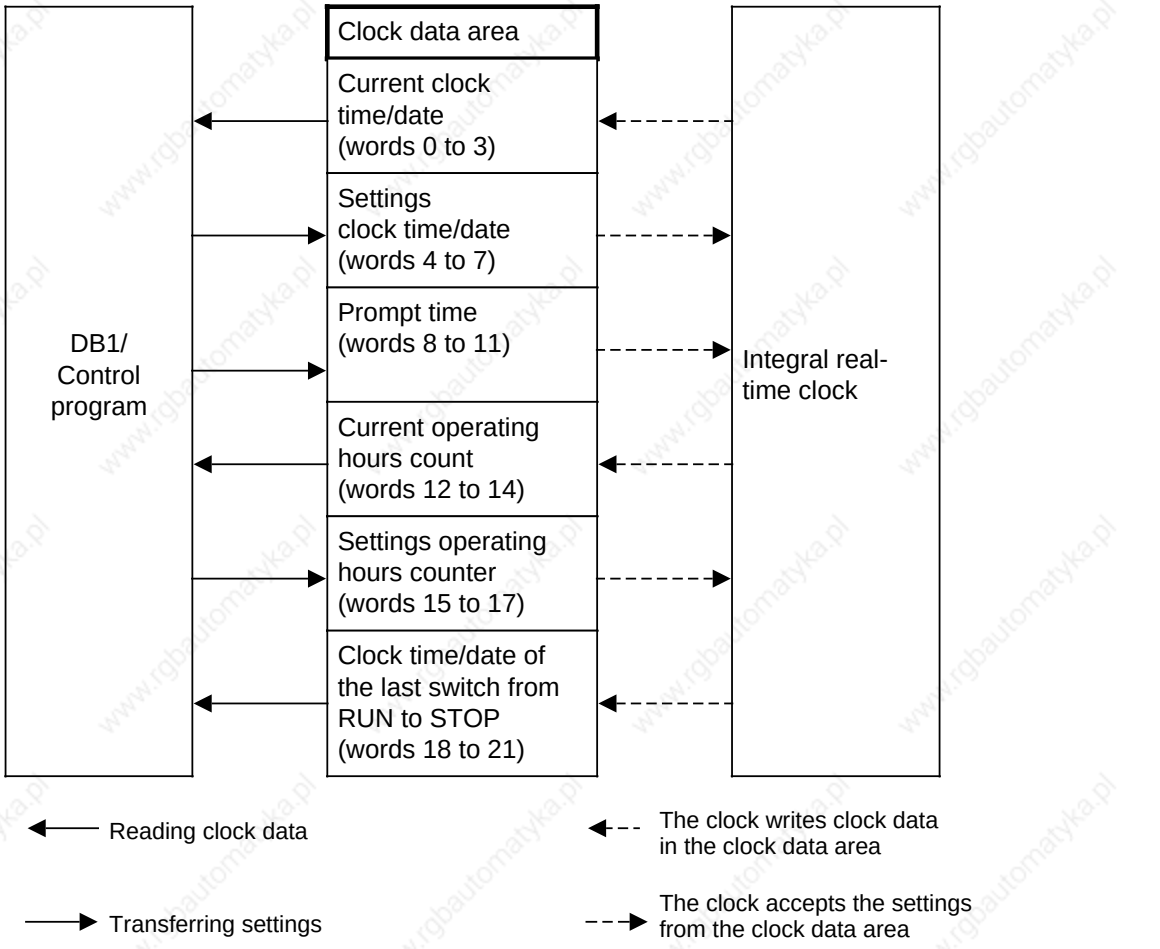



**Figure 12-6. How DB1 or the Control Program and the Clock Access the Clock Data Area**

When you set the clock, you have to transfer only the data needed to implement a particular function. For example, if you want to change only the clock function data, you do not have to enter data for the time prompt function or for the operating hours counter.

The clock data area has the same structure wherever it is located: in the data block area, the flag area, the input area, or the output area. Table 12-3 provides you with information about where specific clock data is located within the clock data area. The explanations for Table 12.3 follow the table.

**Table 12-3. Clock Data in the Clock Data Area**

| Clock Data Area (Data Word) | Meaning | Left Flag Word | Right Flag Word |
|---|---|---|---|
| 0 | Current clock time/date | --- | Weekday |
| 1 | | Day | Month |
| 2 | | Year | AM/PM (Bit 7) Hour |
| 3 | | Minute | Second |
| 4 | Settings for clock time/date | Leap year* | Weekday |
| 5 | | Day | Month |
| 6 | | Year | AM/PM (Bit 7)** Hour |
| 7 | | Minute | Second |
| 8 | Time prompt | --- | Weekday |
| 9 | | Day | Month |
| 10 | | --- | AM/PM (Bit 7)**Hour |
| 11 | | Minute | Second |
| 12 | Current operating hours | --- | Second |
| 13 | | Minutes | Hour |
| 14 | | Hours · 100 | Hours · 10,000 |
| 15 | Settings for operating hours counter | --- | Second |
| 16 | | Minutes | Hours |
| 17 | | Hours · 100 | Hours · 10,000 |
| 18 | Clock time/date after a switch from RUN to STOP or Power OFF (only if bit 5 in the status word = 1) | --- | Weekday |
| 19 | | Day | Month |
| 20 | | Year | AM/PM (Bit 7)** Hour |
| 21 | | Minute | Second |

* Relevant when programming the clock in the user program (see Table 12-4)
** Significant only in the 12-hour mode Bit 7=1 means PM, bit 7=0 means AM

Make certain you are aware of the following points when you make inputs into the clock data area.

- Entries into the clock data area must be in BCD code.
- The clock runs either in the 12-hour mode or the 24-hour mode depending on how you set bit 1 in the status word. See section 12-5 for additional information.
- The AM/PM flag (0 = AM, 1 = PM) is significant only for the 12-hour mode of the hardware clock. The AM/PM flag corresponds to bit 7 in words 2, 6, 10, and 20.
  - In the 12-hour mode, you have to set the hours and the AM/PM flag for both the clock and prompt functions.
  - In the 24-hour mode, if you set an AM/PM flag when you enter the values for the clock and prompting time, then the program sets the relevant error bit.
- Leap year settings are only relevant when the clock is programmed in the user program. If DB1 is used, the leap year settings are carried out automatically by the system.
- The clock settings you enter must be within the range defined in Table 12-4.

**Table 12-4. Range Definitions for Clock Data**

| Variable | Permissible Parameters | Variable | Permissible Parameters |
|---|---|---|---|
| Seconds | 0 to 59 | Day | 1 to 31 |
| Minutes | 0 to 59 | Month | 1 to 12 |
| Hours | In the 24-hour mode:<br>0 to 23<br>In the 12-hour mode:<br>for AM: 1 to12<br>(12 = 12 o'clock noon)<br>for PM: 81 to 92<br>(81 = 1 o'clock PM)<br>0 to 999999 when entering the operating hours | Year | 0 to 99 |
| | | Leap Year | 0 to 3<br>0 = Leap year is current year<br>1 = Leap year was last year<br>2 = Leap year was two years ago<br>3 = Leap year was three years ago |
| Weekday | 1 to 7<br>1=Sunday<br>2=Monday<br>3=Tuesday<br>4=Wednesday<br>5=Thursday<br>6=Friday<br>7=Saturday | | |

If your inputs differ from the ones described, the operating system outputs error messages that are displayed in the status word. The operating system resets error messages displayed in the status word the next time you set the clock, prompt time, or the operating hours counter, if the new settings are within the definition range. See section 12.5.

If you do not wish to modify one of the setting values, you can enter at its place XX (ASCII code) in DB1, or FF (hexadecimal code) when you are programming in the system data.

If the clock data area is located at the end of other areas (flags, data blocks, inputs, and outputs) and there is insufficient memory space available for the clock data area, the amount of clock data transferred is only as much as will fit in the area available. Settings are not accepted if they lie outside of the available space.

- If clock data is located in the non-retentive flag area, then the following two events occur:
  - All the settings are lost after Power OFF and cold restart.
  - The time the last switch from RUN to STOP occurred is lost.

- Remember that you can decide where to locate the clock data area. The word numbers listed in Table 12-3 are relative.
  - If your clock data area is located in a data block and does not begin with data word DW0 but DWX, then you must add the value X to the word number shown in Table 12-3.

    **Example:** Your clock data area begins with DW124. The data for the time and date is then stored in DW124 to DW127.
  - If you locate your clock data area in the flag area beginning with flag word 0, then you must multiply the appropriate data word number listed in Table 12-3 by a factor of 2 to obtain the appropriate flag word address.

    **Example:** You locate your clock data area in the flag operand area beginning with flag word 0. Data for the operating hours counter is then stored beginning with the FW24 address.
  - If your clock data area does not begin at flag word 0, you have to add the beginning value to the word number shown in Table 12-3.

## 12.5 Structure of the Status Word and How to Scan it

You can scan the status word to identify errors in the entered settings. You can deliberately change certain bits in the status word to enable or disable transfer or read operations. You can use designated flag bits to govern the clock's behavior when the programmable controller is switched from the RUN to the STOP mode or during Power OFF.

- The status word can be located in the flag area or in a data block. You must define the location of the status word in DB1 or directly in system data 9 and 10. See section 12.6.

- The integral real-time clock runs independently of the set operating mode. Access to the clock data area depends on the set operating mode and the signal states of bits 4 and 5 in the status word. You can set or reset these bits using the "S" or "R" operations in the control program.
  - If you use an operator panel, such as the OP 396, to monitor the program, it is an advantage to have the programmable controller update the clock time (the current date) even in the STOP mode.

- The operating system resets the "transfer settings" bits (bits 2, 10, and 14 in the status word) under the following conditions.
  - The settings have been transferred.
  - The settings have not been transferred because they were outside of the permissible range. The corresponding error bits (bits 0, 8, and 12 in the status word) are set.

- The operating system does not reset the "transfer settings" bits (bits 2, 10, and 14 in the status word) under the following conditions.
  - The system data for the clock is either incorrect or not available.
  - The clock data area is too small.
  - The clock is defective (hardware error).

- There are four types of bits in the status word.
  - Clock flags
  - Operating system flags
  - Operating hours counter flags
  - Prompt time flags

Tables 12-5 through 12-8 provide you with information about the significance of the signal states of the respective flags.

**Clock Flags**

**Table 12-5. Significance of Bits 0, 1, 2 and 3 of the Status Word**

| Bit Number | Signal State | Meaning |
|---|---|---|
| 0 | 1 | Error in setting entry |
| | 0 | No error in setting entry |
| 1 | 1 | 12-hour clock mode |
| | 0 | 24-hour clock mode |
| 2 | 1 | Transfer settings |
| | 0 | Do not transfer settings |
| 3 | 1 | The clock time can be read |
| | 0 | The clock time cannot be read |

**Operating System Flags**

**Table 12-6. Significance of Bits 4 and 5 of the Status Word**

| Operating Mode | Bit Number Status Word | Signal State | Meaning |
|---|---|---|---|
| STOP | 4 | 1 | The clock updates only words 0 to 3 (current time/date) in the clock data area. You can set the clock by using the FORCE VAR programmer function. |
| | | 0 | The clock does not update the clock data area. Words 0 to 3 contain the time at which the last switch from RUN to STOP occurred. |
| | 5 | 1 | Words 18 to 21 contain the time at which the last RUN to STOP switch occurred or the time at which the last Power OFF occurred if bit 4 is also set. |
| | | 0 | Words 18 to 21 are not used. |
| RUN | 4 | 1/0 | The clock continually updates the clock data area (Words 0 to 17). |
| | 5 | 1 | Words 18 to 21 contain the time at which the last switch from RUN to STOP occurred or the time at which the last Power OFF occurred. |
| | | 0 | Words 18 to 21 are not used. |

### Operating Hours Counter Flags

**Table 12-7. Significance of the Operating Hours Counter Flags Bits 8, 9, and 10 of the Status Word**

| Bit Number | Signal State | Meaning |
|---|---|---|
| 8 | 1 | Error in setting entry |
| | 0 | No error in setting entry |
| 9 | 1 | Enable the operating hours counter |
| | 0 | Disable the operating hours counter |
| 10 | 1 | Transfer the settings |
| | 0 | Do not transfer the settings |

### Prompt Time Flags

**Table 12-8. Significance of the Prompt Time Flag Bits 12, 13, and 14 of the Status Word**

| Bit Number | Signal State | Meaning |
|---|---|---|
| 12 | 1 | Error in setting entry |
| | 0 | No error in setting entry |
| 13 | 1 | The set prompting time is reached |
| | 0 | The set prompting time is not reached |
| 14 | 1 | Transfer the settings |
| | 0 | Do not transfer settings |

The operating system requires bit numbers 6, 7, 11, and 15. You can not use these bits.

### Scanning the Status Word

In a data block, you can use the “P <data word number> <bit number>” operation to scan the individual bits of a data word. In the flag area, you can scan the individual bits if you enter the <byte address> and the <bit number>.

**Example:** The status word is stored in DW13. You are checking to see if the set prompt time has been reached. The “P D 13.13” instruction triggers a scan. If the status word is stored in FW12, then the same scan would be “A F 12.5”.

### Backup of the Hardware Clock

If there is a backup battery, the clock continues to run even after Power OFF. If the programmable controller does not have a backup battery, the clock values will be set at “April 1, 1992, 12.00.00 o’clock, weekday: 4” when the clock is initialized after a Power ON. The default is the 24-hour time mode. You should install a battery only during Power ON; otherwise, you would lose the clock data.

## 12.6 Setting Parameters for the Clock Data Area and the Status Word in the System Data Area

**Table 12-9. The System Data Area for the Integral Real-Time Clock**

| Absolute Address RAM | System Data Word | Meaning | Permissible Parameters |
|---|---|---|---|
| EA10 | 8 | Operand area for the clock data | ASCII characters<br>I, Q, F, D |
| EA11 | | Start address for the clock data<br>Operand area D<br>Operand Areas I, Q, F | DB number<br>DB2 to DBFF$_H$<br>Byte address |
| EA12 | 9 | Start address for the clock data<br>Relevant only for operand area D | DB word number<br>DW0 to DWFF$_H$ |
| EA13 | | Operand area for the status word | ASCII characters<br>I, Q, F, D |
| EA14 | 10 | Start address for the status word<br>Operand area D<br>Operand areas I, Q, F | DB number<br>DB2 to DBFF$_H$<br>Byte address |
| EA15 | | Start address for the status word<br>Relevant only for operand area D | DB word number<br>DW0 to DWFF$_H$ |
| EA16 | 11 | Status for hardware[1]<br>(only bits 0 and 1 are relevant)<br>• If either bit 0 or bit 1 is set, the clock chip is defective<br>• If no bit is set, the clock chip is running | “0”, “1” |
| EA16 | 11 | Incorrect correction value?<br>(only bit 15 is relevant)<br>• If bit 15 is set, the correction value is incorrect (>+400 or <- 400)<br>• If bit 15 is not set, the correction value is correct | “0”, “1” |
| EA18 | 12 | Correction value[2] | - 400 to 400 |

1 You can scan SD11 during start-up. You must call up an FB in OB21 or OB22 by using “L RS 11” to read out and then continue processing SD11.
2 Always use the “L KF X” instruction to load the correction value in ACCU 1 since negative values can also be specified.

The following section is intended to help you to start running the integral real-time clock as quickly as possible by setting parameters in the system data. You need to be familiar with the clock data area described in sections 12.4 and 12.5 in order to understand this section.

> **Note**
>
> The clock time is updated one second after the start of the next cycle.

**Task:**

You want to set the clock to the following values:

Wednesday, 12.02.92; 10:30:00

The status word is assigned to flag word FW12. Clock data is stored in DB75 beginning with DW0. The two ways to transfer clock settings are as follows:

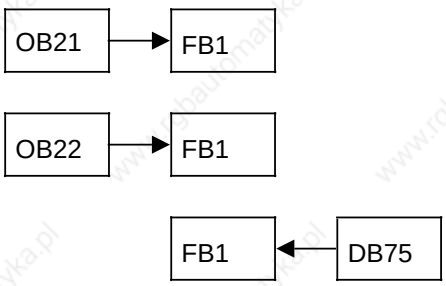
- Use the STATUS VAR programmer function when the programmable controller is in the RUN mode.
- Use the FORCE VAR programmer function when the programmable controller is in the STOP mode and status word bit 4 = 1.

The following example uses the first method. Proceed as follows:

1. Set the programmable controller to Power OFF.
2. Set the operating mode switch to STOP.
3. Set the programmable controller to Power ON.
4. Perform an overall reset on the programmable controller. See section 4.1.3.
5. Use the following program to program the programmable controller.
6. Set the operating mode switch to RUN.

The integral real-time clock is running.

**Program Structure:**




The system data is defined during the change from STOP to RUN.

The system data is defined when the programmable controller is switched on.

Set the new date and time.

**The Block Entry Sequence and a Programming Example:**

The following procedure is suggested:

1. Program FB1 - Defining system data for the integral real-time clock
2. Program OB21 - Calling up FB1 during a change from STOP to RUN
3. Program OB22 - Calling up FB1 when the programmable controller is switched on
4. Generate DB75 - Storing clock data
5. Transfer new data to the clock using the FORCE VAR programmer function (programmable controller in the RUN mode) .

The respective programming examples are shown in Tables 12-10 to 12-14.

**Table 12-10. FB1 Program**

| STL | Meaning | Explanation |
|---|---|---|
| FB 1<br>NAME: CLOCK<br><br>L KH 4 4 4 B<br><br>T RS 8<br><br>L KH 0 0 4 D<br><br>T RS 9<br><br>L KH 0 C 0 0<br><br>T RS 10<br><br>BE | ASCII Code for the “D” character<br><br>Block number “$75_D$”<br>Storing in system data word 8<br>DW0 is the start address for clock data<br><br>ASCII Code for the “M” Character<br>Storing in system data word 9<br>Flag word Number “$12_D$”<br><br>ASCII Code is irrelevant<br>Storing in system data word 10 | Clock data is located in DB75 beginning with DW0<br><br>The status word is located in FW12 |

**Table 12-11. OB21 Program**

| STL | Explanation |
|---|---|
| OB 21<br>JU FB 1<br>NAME: CLOCK<br>BE | The function block is called up once during a switch from STOP to RUN. |

**Table 12-12. OB22 Program**

| STL | Explanation |
|---|---|
| OB 22<br>JU FB 1<br>NAME: CLOCK<br>BE | The function block is called up once when the programmable controller is switched on. |

**Table 12-13. DB75 Program**

| STL | Explanation |
|---|---|
| DB 75<br>0: KH = 0000;<br>1: KH = 0000;<br>2: KH = 0000;<br>3: KH = 0000;<br>4: KH = 0000;<br>5: KH = 0000;<br>6: KH = 0000;<br>7: KH = 0000 | Define the number of data words. (Data words 0 to 7 are used in the example. See Table 12-3) . Define the numerical representation. Hex is used in the examples. |

**Reading and Setting the Time and Date**

After you enter the program, you can test it as follows.

1. Switch the programmable controller to the RUN mode.
2. Use the FORCE VAR programmer function to enter the following.
   - Data block number
   - Data words DW0 to DW7
   - Clock data
   - Status word

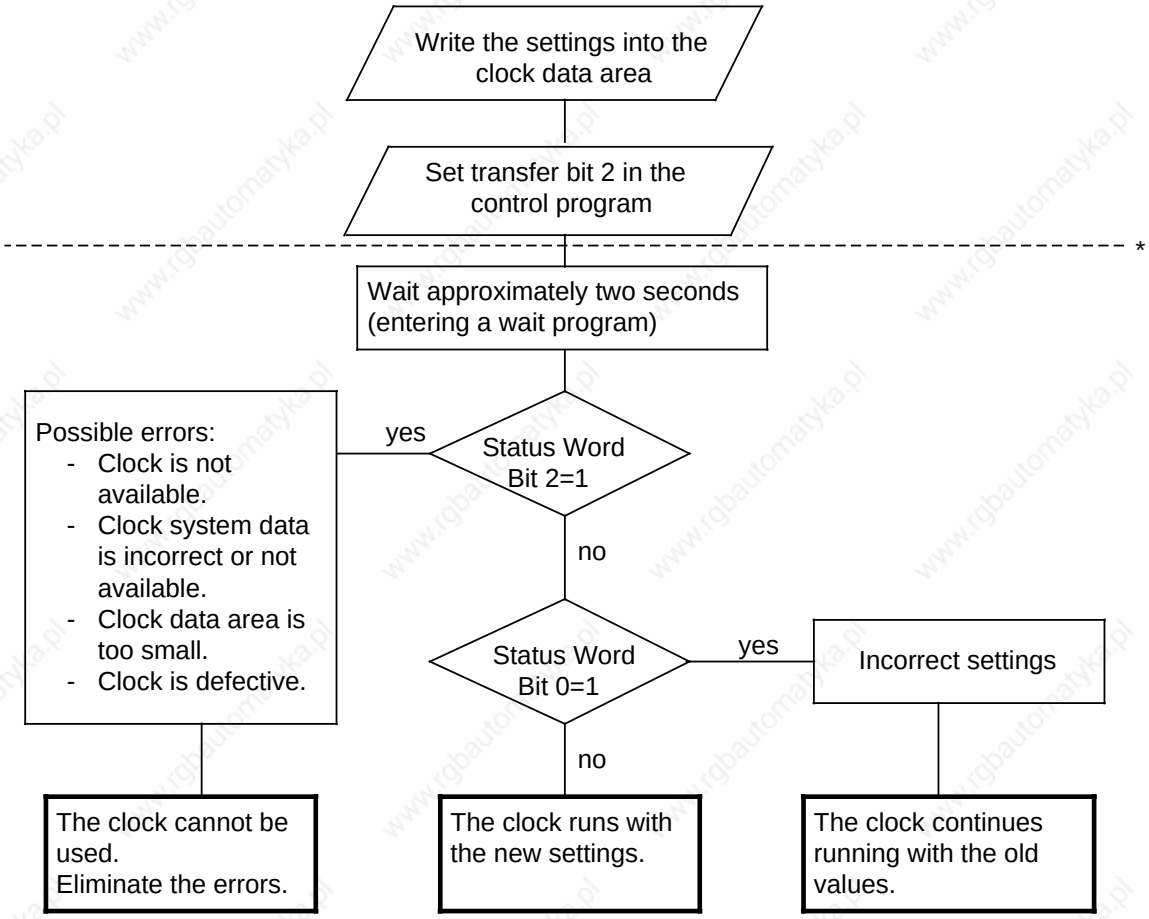
**Table 12-14. FORCE VAR Function**

| Operand | Signal States | Explanation |
|---|---|---|
| DB 75 | | |
| DW 0 | KH = 0003 | Tuesday |
| DW 1 | KH = 0110 | October 1 |
| DW 2 | KH = 9112 | 1991, 12 o'clock |
| DW 3 | KH = 0000 | (reading current clock data) |
| DW 4 | KH = 0002 | Monday |
| DW 5 | KH = 0212 | December 2 |
| DW 6 | KH = 9210 | 1992, 10.30 o'clock |
| DW 7 | KH = 3000 | (writing new settings) |
| FW 12 | KM = 00000000 00000**1**00 | If you set bit 2 in the status word to "1", the new settings are transferred to the clock. |

3. Begin status processing by pressing the ENTER key twice. Bit 2 in the status word is reset. The clock runs with the new settings

**Note**

Besides using the method described in Table 12-14 (with the FORCE VAR function), you can also enter the new settings directly into the data block. In that case, store the new settings in data words DW4 to DW7 of data block DB75.
See Table 12-13.




* The lower portion of the flowchart has only a diagnostic function. There is nothing you must perform. You can also implement the upper portion of the flowchart using the FORCE VAR programmer function (programmable controller in the RUN mode) or using the FORCE function (the programmable controller is in the STOP mode and bit 4 = 1 in the status word).

**Figure 12-7. Flowchart - Transferring Time and Date Settings to the Clock**

If you do not want a value (for example the minutes) in the settings to be transferred, enter the value for relevant byte as either $255_D$ or $FF_H$. When you set the clock, the old value present in the clock is retained.

Incorrect settings are displayed by a set bit 0 in the status word. The clock continues to run with the old values.

In a similar manner, you can program new settings for the time prompt function and the operating hours counter. However, the settings are located in other data words in the clock data area. See section 12-4. You must set the respective bit to 1 in the status word so that the clock can accept the new settings. See section 12-5.

## 12.7 Programming the Integral Real-Time Clock in the User Program

The programming of the clock in the user program should be performed only by users with extensive knowledge of the system. For all other users, use of DB1 is recommended (see sections 12.2 and 12.3).

The following section provides you with information on how to access the clock through the user program.

### 12.7.1 Reading and Setting the Clock

**Example:** Program for setting the time and date

Transfer of the settings for the time and date is triggered by input I 0.0. Before you set input I 0.0 (see OB1), you must transfer these settings to flag bytes FY120 to FY127. Values that you do not want to change must be preset with "$FF_H$". You can define the clock mode with input I 1.0(1= 12-hour mode). Input I 0.1 is the AM/PM bit that you use for setting the 12-hour mode.

The clock data area is in DB2 beginning with DW0, and the status word is FW10.

| OB1 STL | Explanation |
|---|---|
| : | ================================ |
| : | SETTING THE TIME AND DATE |
| : | ================================ |
| : | FIRST TRANSFER TIME AND DATE VALUES |
| : | INTO FB120 TO FB127. |
| :A I 0.0 | CLOCK SETTING TRIGGERED |
| :S F 20.0 | BY SETTING F 20.0 (RESET IN FB10) |
| :JU FB 10 | |
| NAME :SET CLOCK | (SETTING THE TIME AND DATE) |
| WDAY : FY 121 | WEEKDAY |
| DAY : FY 122 | DAY |
| MON : FY 123 | MONTH |
| YEAR : FY 124 | YEAR |
| HOUR : FY 125 | HOUR |
| AMPM : I 0.1 | AMPM-BIT (ONLY IMPORTANT IN 12-HOUR MODE) |
| MIN : FY 126 | MINUTES |
| SEC : FY 127 | SECONDS |
| ERR : F 12.1 | ERROR BIT |
| MODE : I 1.0 | 12-HOUR MODE: I 1.0 = 1 |
| :BE | |

| FB10 STL | Description |
|---|---|
| NAME :SET CLOCK | SETTING THE CLOCK |
| DES :WDAY I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :DAY I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :MON I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :YEAR I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :HOUR I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :AMPM I/Q/D/B/T/C: I BI/BY/W/D: BI | |
| DES :MIN I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :SEC I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :ERR I/Q/D/B/T/C: Q BI/BY/W/D: BI | |
| DES :MODE I/Q/D/B/T/C: I BI/BY/W/D: BI | |
| :A =MODE | 24HR-MODE = 0, 12HR-MODE = 1 |
| := F 11.1 | (CLOCK MODE STATUS WORD BIT 1) |
| :AN F 20.0 | FLAG IS RESET IF SETTINGS ALREADY |
| :JC =M001 | READ INTO CLOCK DATA AREA |
| :R F 20.0 | |
| : | |
| :C DB 2 | CLOCK DATA AREA |
| :L =WDAY | STORE VALUE FOR WEEKDAY |
| :T DR 4 | |
| :L =DAY | STORE VALUE FOR DAY |
| :T DR 5 | |
| :L =MON | STORE VALUE FOR MONTH |
| :T DR 5 | |
| :L = YEAR | STORE VALUE FOR YEAR |
| :T DL 6 | |
| :L =HOUR | STORE VALUE FOR HOUR |
| :ON =AMPM | IF 12-HOUR MODE IS SET, AND |
| :ON =MODE | AM/PM BIT = 1 (AFTERNOON), THE |
| :JC =MORN | RELEVANT BIT IN THE CLOCK AREA |
| :L KH 0080 | IS SET |
| :OW | |
| MORN :T DR 6 | |
| :L =MIN | STORE VALUE FOR MINUTES |
| :T DL 7 | |
| :L =SEC | STORE VALUE FOR SECONDS |
| :T DR 7 | |
| :AN F 11.2 | TRANSFER SETTINGS |
| :S F 11.2 | (STATUS WORD IS FW10) |
| :L KT 020.1 | START MONITORING TIME |
| :SE T 10 | |
| M001 :A T 10 | BEC, IF MONITORING TIME |
| :BEC | NOT YET ELAPSED |

| FB10 STL (continued) | Explanation |
|---|---|
| :AN F 11.2 | HAVE SETTINGS BEEN TRANSFERRED? |
| :JC =M002 | IF YES, JUMP TO M002 |
| :S =ERR | SET ERROR BIT IF THERE ARE ERRORS |
| :BEU | |
| M002 :AN F 11.0 | WERE THERE ERRORS WHILE ENTERING SETTINGS? |
| :RB =ERR | IF NO, RESET ERROR BIT |
| :BEC | IF NO ERROR, THEN BEC |
| :S =ERR | IF AN ERROR, SET ERROR BIT |
| :BE | |

**Example:** Program for reading the current time and the current date

The time is stored in flag bytes FY30 to FY36, depending on an external event, simulated here by a positive edge at input I 0.5. Flag F 13.1 indicates which mode the clock is operating in. Flag F 13.0 is the AM/PM bit in the 12-hour mode

The clock data area is in DB2 beginning with DW0, and the status word is FW10.

| OB1 STL | Explanation |
|---|---|
| : | ================================ |
| : | READING TIME AND DATE |
| : | ================================ |
| :A I 0.5 | TIME AND DATE ARE |
| :AN F 0.1 | STORED IN FY30 TO FY36 IN CASE OF A |
| := F 0.0 | POSITIVE EDGE AT I 32.5. |
| :A I 32.5 | (EXTERNAL EVENT) |
| := F 0.1 | |
| : | |
| :A F 0.0 | EDGE TRIGGER FLAG |
| :JC FB 13 | |
| NAME :READ CLOCK | (READING TIME AND DATE) |
| WDAY : FY 30 | WEEKDAY |
| DAY : FY 31 | DAY |
| MON : FY 32 | MONTH |
| YEAR : FY 33 | YEAR |
| HOUR : FY 34 | HOUR |
| AMPM : F 13.0 | F 13.0 = 1, AFTERNOON IN 12H-MODE |
| MIN : FY 35 | MINUTES |
| SEC : FY 36 | SECONDS |
| MODE : F 13.1 | F 13.1 = 1, IN 12-HOUR MODE |
| :BE | |

| FB13 STL | Explanation |
|---|---|
| NAME :READ CLOCK | READING THE CLOCK |
| DES :WDAY I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :DAY I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :MON I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :YEAR I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :HOUR I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :AMPM I/Q/D/B/T/C: Q BI/BY/W/D/:BI | |
| DES :MIN I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :SEC I/Q/D/B/T/C: Q BI/BY/W/D/:BY | |
| DES :MODE I/Q/D/B/T/C: Q BI/BY/W/D/:BI | |
| | |
| :C DB 2 | |
| :L DR 0 | WEEKDAY |
| :T =WDAY | |
| :L DL 1 | DAY |
| :T =DAY | |
| :L DR 1 | MONTH |
| :T =MON | |
| :L DL 2 | YEAR |
| :T =YEAR | |
| :L DR 2 | HOUR |
| :L KH 007F | ERASE AM/PM BIT |
| :AW | (ONLY RELEVANT IN 12-HOUR MODE) |
| :T =HOUR | |
| :TB D 2.7 | DISPLAY AM/PM BIT |
| := =AMPM | (ONLY RELEVANT IN 12-HOUR MODE) |
| :L DL 3 | MINUTE |
| :T =MIN | |
| :L DR 3 | SECOND |
| :T =SEC | |
| :A F 11.1 | DISPLAY CLOCK MODE |
| := =MODE | MODE = 1, IN 12-HOUR MODE |
| :BE | |

**Storing the Updated Time/Date after a RUN to STOP Switch**

> **Note**
>
> This clock data area is only written to if the following requirements are met.
> - Bit 5 in the status word is set to “1”.
> - A RUN to STOP switch or a Power OFF has taken place.
> - The necessary memory space is available in the operand area.

This enables you to detect a RUN to STOP switch or a Power OFF even if the programmable controller has since gone back to RUN mode. The time and date of the last RUN to STOP switch or Power OFF are in words 18 to 21 (see Table 12-3)

If several RUN to STOP switches have occurred before you read out this clock data area, you will only be able to determine the time of the last switch.

If you do not have sufficient memory for this clock data area, you either cannot use this area or use only part of it. This has no impact on anything else.

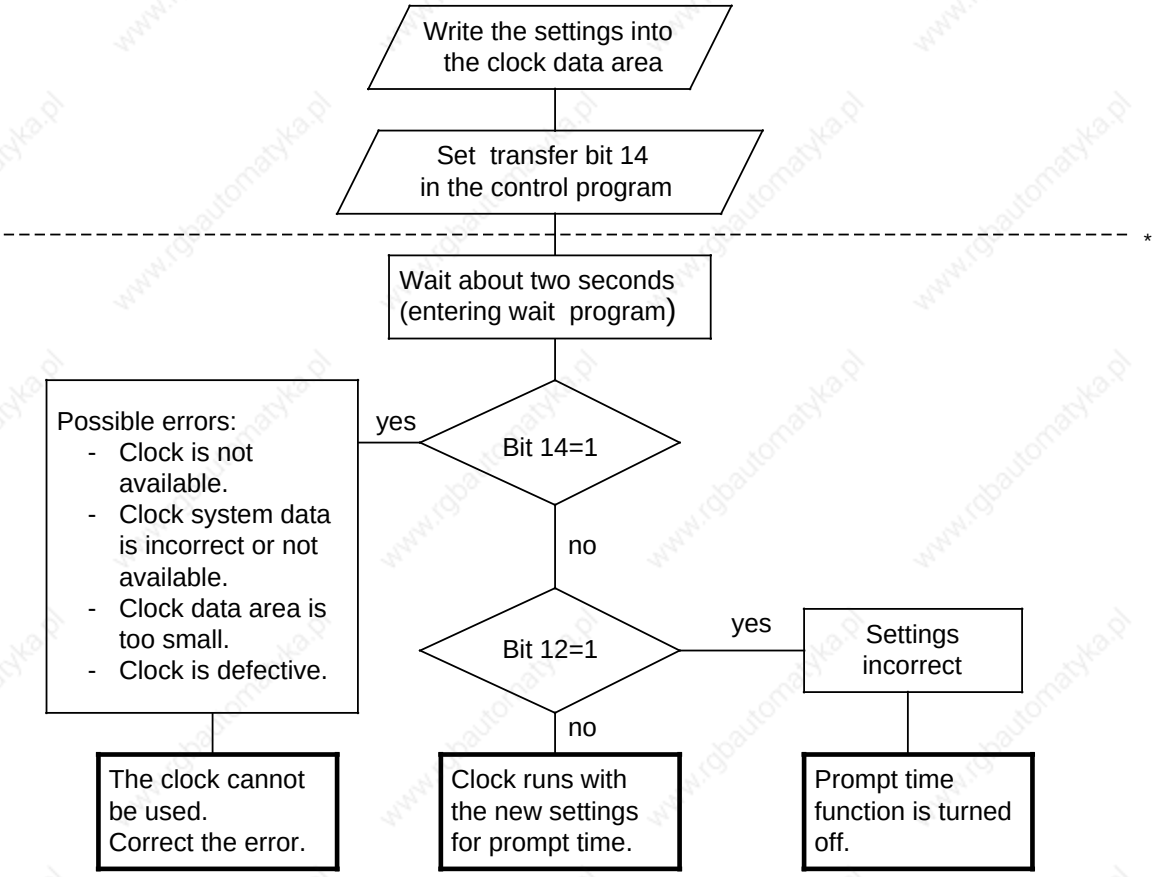
## 12.7.2 Programming the Prompt Function

**Transferring Settings to the Clock**

- You can store the settings in the clock data area by using transfer operations (see Table 12-3).
- The AM/PM flag (bit number 7) is only significant in 12-hour mode.
  Bit 7=1 means PM
  Bit 7=0 means AM
- You must transfer the clock data in BCD code.

  TIP: The “KC” data format loads a BCD constant into ACCU 1 and is therefore especially suitable.

- If you enter the value “$255_D$” or “$FF_H$” in a byte as the prompt time, this byte will be ignored when evaluating “Prompt time reached”. This makes it easy to program, for example, an alarm that is repeated daily by entering the value in the “$255_D$” or “$FF_H$”in the “Weekday”, “Date” and “Month” settings.
- You can transfer the prompt time settings to the clock by initiating bit 14 in the status word.
- The settings are transferred 1 second after the start of the next cycle.
- Bit 12 in the status word displays incorrect settings.




* The lower part of the flow chart has only a diagnostic function. There is nothing you must perform.

**Figure 12-8. Flowchart - Transferring a New Prompt Time**

**Prompt Time Sequence**

- Bit 13 in the status word is set after the prompt time has elapsed.
- Bit 13 remains set until you reset it in the control program.
- The prompt time can be read at any time.

> **Caution**
>
> If the prompt time is reached in the STOP mode or during Power OFF, the prompt time cannot be evaluated. It is always deleted on restart.

**Example:** Setting and evaluating the prompt time

In the example program, the status of input I 0.6 determines whether the settings for the prompt time are transferred. Before setting input I 0.6, you must transfer the settings to flag bytes FY130 and FY135. Enter values that you do not wish to be evaluated as $FF_H$.

You set the clock mode with input I 1.0. Use input I 0.1 to specify the the AM/PM bit for 12-hour mode.

If the preset prompt time has been reached, set flag F 13.2. If errors are made while entering the prompt time, the error bit, flag F 12.2, is set.

The clock data is stored in DB2 beginning with data word DW0, and the status word is flag word FW10.

| OB1 STL | Explanation |
|---|---|
| : | ========== |
| : | SETTING AND EVALUATING THE PROMPT TIME |
| : | ========== |
| : | LOAD VALUES INTO FY130 TO FY135 |
| : | FIRST. |
| :A I 0.6 | TRIGGER SETTING OF PROMPT TIME |
| :S F 20.1 | BY SETTING F 20.1 (RESET IN FB11) |
| :JU FB 11 | |
| NAME :SET PROMPT TIME | (SET AND EVALUATING PROMPT TIME) |
| WDAY : FY 130 | WEEKDAY |
| DAY : FY 131 | DAY |
| MON : FY 132 | MONTH |
| HOUR : FY 133 | HOUR |
| AMPM : I 0.1 | AMPM-BIT (ONLY IMPORTANT IN 12-HOUR MODE) |
| MIN : FY 134 | MINUTES |
| SEC : FY 135 | SECONDS |
| ERR : F 12.2 | ERROR BIT |
| ALRM : F 13.2 | DISPLAYS THAT PROMPT TIME IS REACHED. |
| MODE : I 1.0 | 12-HOUR MODE: I 33.0 = 1 |
| :BE | |

| FB11 STL | Explanation |
|---|---|
| NAME :SET PROMPT TIME | SETTING THE PROMPT TIME |
| DES :WDAY I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :DATE I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :MON I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :HOUR I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :AMPM I/Q/D/B/T/C: I BI/BY/W/D: BI | |
| DES :MIN I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :SEC I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :ERR I/Q/D/B/T/C: Q BI/BY/W/D: BI | |
| DES :ALRM I/Q/D/B/T/C: Q BI/BY/W/D: BI | |
| DES :MODE I/Q/D/B/T/C: I BI/BY/W/D: BI | |
| :A =MODE | 24 HOUR MODE = 0, 12 HOUR = 1 |
| := F 11.1 | (SET CLOCK MODE) |
| :A F 10.5 | DISPLAY PROMPT TIME REACHED |
| :S =ALRM | (BIT 13 IN STATUS WORD) |
| :R F 10.5 | RESET BIT AFTER EVALUATION |
| : | |
| :AN F 20.1 | FLAG IS RESET IF SETTINGS HAVE ALREADY |
| :JC =M001 | BEEN READ INTO THE CLOCK DATA AREA |
| :R F 20.1 | |
| : | |
| :C DB 2 | CLOCK DATA AREA |
| :L =WDAY | STORE VALUE FOR WEEKDAY |
| :T DR 8 | |
| :L =DAY | STORE VALUE FOR DATE |
| :T DL 9 | |
| :L =MON | STORE VALUE FOR MONTH |
| :T DR 9 | |

| FB11 STL (continued) | Description |
|---|---|
| :L =HOUR | STORE VALUE FOR HOURS |
| :ON =AMPM | IF AM/PM = 1 (AFTERNOON) AND |
| :ON =MODE | 12-HOUR MODE IS SET, THE |
| :JC =MORN | CORRESPONDING BIT IN THE CLOCK |
| :L KH 0080 | DATA AREA IS SET |
| :OW | |
| MORN :T DR 10 | |
| :L =MIN | STORE VALUE FOR MINUTES |
| :T DL 11 | |
| :L =SEC | STORE VALUE FOR SECONDS |
| :T DR 11 | |
| :AN F 10.6 | TRANSFER SETTINGS |
| :S F 10.6 | (BIT 14 IN STATUS WORD FW10) |
| :L KT 020.1 | START MONITORING TIME |
| :SE T 11 | |
| M001 :A T 11 | BEC, IF MONITORING TIME NOT YET |
| :BEC | ELAPSED |
| :AN F 10.6 | HAVE SETTINGS BEEN TRANSFERRED? |
| :JC =M002 | IF YES, JUMP TO M002 |
| :S =ERR | IF ERROR, SET ERROR BIT |
| :BEU | |
| M002 :AN F 10.4 | ERROR WHEN ENTERING SETTINGS? |
| :RB =ERR | IF NO, RESET ERROR BIT |
| :BEC | BEC, IF NO ERROR |
| :S =ERR | IF ERROR, SET ERROR BIT |
| :BE | |

### 12.7.3 Programming the Operating Hours Counter

You can enable the operating hours counter with bit 9 of the status word. This allows you to establish, for example, the number of hours a motor has been in operation. The operating hours counter is active only in the RUN mode.

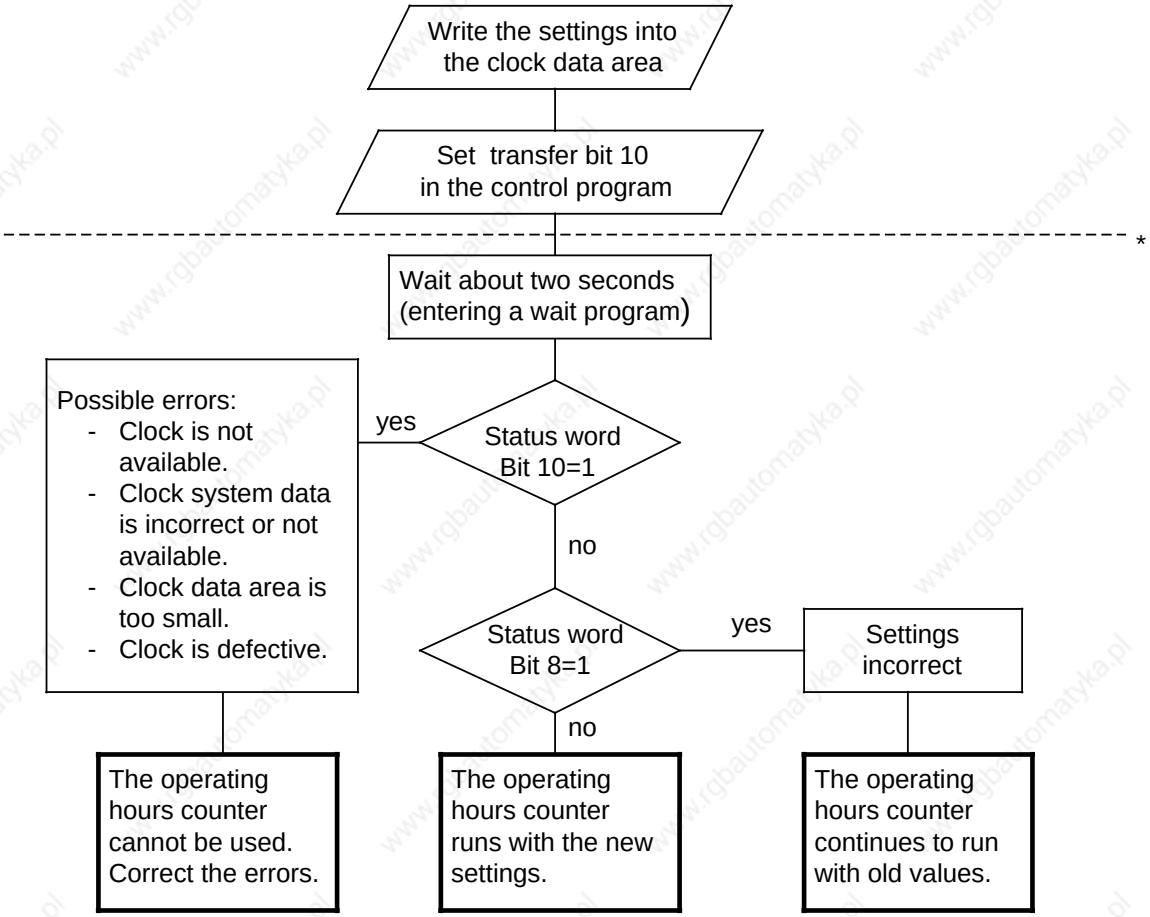
#### Transferring Settings to the Operating Hours Counter

You can preset the operating hours counter with a certain start value (e.g. after exchanging the CPU).

- The clock data must be transferred in BCD code.

  TIP: The “KC” data format loads a BCD constant into ACCU 1 and is therefore especially suitable for entering the settings.

- If you do not want a value (for example minutes) to be transferred, entering the relevant byte as “$255_D$” or “$FF_H$”. The current value for this variable is then retained.
- After you have transferred the settings to the clock data area, you must set bit 10 in the status word for the clock to accept the clock data.
- Bit 8 in the status word displays incorrect settings.




* The lower part of the flow chart has only a diagnostic function. There is nothing you must perform.

**Figure 12-9. Flowchart - Transferring Settings to the Operating Hours Counter**

**Example:** Setting the operating hours counter

The status of input I 0.7 determines whether the operating hours counter values are transferred. You must transfer these values to flag bytes FY136 to FY140 before setting input I 0.7 (not implemented in the example program). Values that are not to be changed should be preset with $FF_H$.

Errors are displayed in flag F 12.3. The clock data area is in data block DB2 beginning with data word DW0, and the status word is flag word FW10.

| OB1 STL | Explanation |
|---|---|
| :<br>:<br>:<br>:<br>:<br>:A I 0.7<br>:S F 20.2<br>:<br>:JU FB 12<br>NAME :SET OPER. HOURS COUNTER<br>SEC : FY 136<br>MIN : FY 137<br>HOUR0: FY 138<br>HOUR2: FY 139<br>HOUR4: FY 140<br>ERR : F 12.3<br>:BE | ==========<br>SETTING THE OPERATING HOURS COUNTER<br>==========<br>LOAD VALUES INTO FY136 TO FY140<br><br>TRIGGER TRANSFER OF SETTINGS FOR<br>OPERATING HOURS COUNTER BY SETTING<br>F 20.2<br><br>(SETTING THE OPERATING HOURS COUNTER)<br>SECONDS<br>MINUTES<br>HOURS<br>HOURS X 100<br>ERRORS X 10000<br>ERROR BIT |

| FB12 STL | Explanation |
|---|---|
| NAME :SET OPER. HOURS COUNTER | SETTING THE OPERATING HOURS COUNTER |
| DES :SEC I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :MIN I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :HOUR0 I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :HOUR2 I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :HOUR4 I/Q/D/B/T/C: I BI/BY/W/D: BY | |
| DES :ERR I/Q/D/B/T/C: Q BI/BY/W/D: BI | |
| :AN F 20.2 | FLAG IS RESET IF SETTINGS |
| :JC =M001 | ALREADY READ INTO THE |
| :R F 20.2 | CLOCK DATA AREA |
| : | |
| :C DB 2 | CLOCK DATA AREA |
| :L =SEC | STORE VALUE FOR SECONDS |
| :T DR 15 | |
| :L =MIN | STORE VALUE FOR MINUTES |
| :T DL 16 | |
| :L =HOUR0 | STORE VALUE FOR HOURS |
| :T DR 16 | |
| :L =HOUR2 | STORE VALUE FOR HOURS X 100 |
| :T DL 17 | |
| :L =HOUR4 | STORE VALUE FOR HOURS X 1000 |
| :T DR 17 | |
| :AN F 10.2 | TRANSFER SETTINGS |
| :S F 10.2 | (BIT 10 IN STATUS WORD FW 10) |
| :S F 10.1 | ENABLE OPERATING HOURS COUNTER |
| : | IF NOT ALREADY ENABLED |
| :L KT 020.1 | START MONITORING TIME |
| :SE T 12 | |
| M001 :A T 12 | BEC IF MONITORING TIME NOT YET |
| :BEC | ELAPSED |
| :AN F 10.2 | HAVE SETTINGS BEEN TRANSFERRED? |
| :JC =M002 | IF YES, JUMP TO M002 |
| :S =ERR | IF ERROR, SET ERROR BIT |
| :BEU | |
| M002 :AN F 10.0 | ERROR WHEN ENTERING SETTINGS? |
| :RB =ERR | IF NO, RESET ERROR BIT |
| :BEC | BEC IF NO ERROR |
| :S =ERR | IF ERROR, SET ERROR BIT |
| :BE | |

### Reading the Current Operating Hours Counter

The current data is stored in words 12 to 14 of the clock data area. You can use load operations to read out the data.

**Example:** Reading the operating hours counter

You need to switch off a machine for inspection after every 300 hours of operation. Flag F 12.4 is set when the machine is switched off. After 300 hours of operation, a jump is made to PB5 to switch the machine off (not programmed in the example).
The clock data area is in DB2 beginning with flag word FW0, and the status word is flag word FW10.

| OB1 STL | Explanation |
|---|---|
| :JU FB 14<br>NAME :BETR-LES<br>:<br>:BE | EVALUATE OPERATING HOURS COUNTER |

| FB14 STL | Explanation |
|---|---|
| NAME :BETR-LES<br><br>:C DB 2<br>:A F 12.4<br>:BEC<br>:<br>:L DL 14<br>:<br>:L KC 003<br>:><F<br>:BEC<br>:<br>:S F 12.4<br>:JU PB 5<br>:<br>:<br>:BE | READING THE OPERATING HOURS COUNTER<br><br>DB IN WHICH THE CLOCK DATA IS LOCATED.<br>IF AUXILIARY FLAG 12.4 IS SET, OFF.<br>THE MACHINE IS ALREADY<br>BLOCK END<br>LOAD HOUR VALUE X 100 IN ACCU 1<br><br>COMPARE TO 3 (=300 HOURS)<br><br>END IF 300 HOURS NOT YET<br>REACHED<br>SET AUXILIARY FLAG<br>JUMP TO PB5 WHEN 300 OPERATING<br>HOURS REACHED |

### 12.7.4 Entering the Clock Time Correction Factor

You can configure a correction value that increases the exactness of the integral real-time clock. The correction value is displayed in seconds/month. The month is defined as 30 days.

| Absolute Address RAM Memory | Range | System Data Word |
|---|---|---|
| EA 18 | $-400_D$ to $+400_D$ seconds/month | 12 |

**Example:** You determined that the clock runs 12 seconds slow in a four day period. That would be 90 seconds in 30 days. The correction value is+ 90 seconds/month.

**Note**

Use the data KF format to enter the correction value. You then do not have to convert the value to other numbering systems.

| STL | Explanation |
|---|---|
| FB10<br>L KF + 90<br>T RS 12<br>BE | LOAD THE + 90 SECONDS CORRECTION VALUE INTO ACCU 1 AND STORE IT IN SYSTEM DATA WORD 12. |

**Note**

The correction value you have entered is read in after the next minute change. If an error occurs when a setting is entered, bit 15 in system data word 11 is set.

## 13 Connecting the S5-100U to SINEC L1, for CPU 102 and Higher


13.1 Connecting the Programmable Controllers to the L1 Bus Cable . . . . . 13 - 1

13.2 Setting Parameters in the Programmable Controller for Exchanging Data . . . . . 13 - 1
13.2.1 How to Program in a Function Block, for CPU 102 and Higher . . . . 13 - 2
13.2.2 Setting Parameters in DB1, for CPU 103 and Higher . . . . . 13 - 5

13.3 Coordinating Data Exchange in the Control Program . . . . . 13 - 7
13.3.1 Sending Data . . . . . 13 - 8
13.3.2 Receiving Data . . . . . 13 - 9
13.3.3 Programming the Messages in a Function Block . . . . . 13 - 11

**Figures**


13-1 Connection of the Bus Cable . . . . . 13 - 1
13-2 Programming Example for Setting Parameters in FB1 . . . . . 13 - 4
13-3 Data Exchange between Sender and Receiver (Principle) . . . . . 13 - 7
13-4 Structure of the Send Mailbox . . . . . 13 - 8
13-5 Structure of the Coordination Byte Send (KBS) . . . . . 13 - 8
13-6 Structure of the Receive Mailbox . . . . . 13 - 9
13-7 Structure of the Coordination Byte Receive (KBE) . . . . . 13 - 10
13-8 Organization of Program Execution . . . . . 13 - 11
13-9 Programming “Message Processing” in FB2 . . . . . 13 - 12


**Tables**


13-1 SINEC L1 Parameter Block . . . . . 13 - 2
13-2 Setting Parameters in the Coordination Byte . . . . . 13 - 3
13-3 Setting Parameters for the SINEC L1 Interface . . . . . 13 - 6

# 13 Connecting the S5-100U to SINEC L1, for CPU 102 and Higher

SINEC L1 is a local area network that enables SIMATIC S5 programmable controllers to communicate with each other. This option is available when you are using CPU 102 or higher. It operates on the master-slave principle.

You will find more exact information on the SINEC L1 in the *SINEC L1* manual. You need to understand the SINEC L1 operating system before continuing with this chapter.

The S5-100U can be connected directly to the SINEC L1 as a slave. The information you need to perform this operation is explained in this chapter.

## 13.1 Connecting the Programmable Controller to the L1 Bus Cable

Bus terminal BT 777 is the signal level converter that connects the programmable controller to the L1 bus cable. The procedure is as follows:

1. Connect the L1 bus cable to bus terminal BT 777.

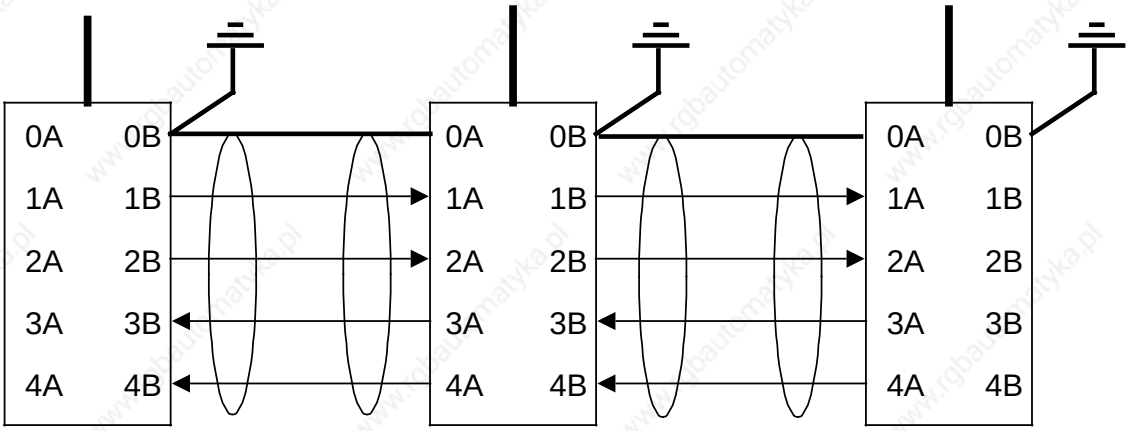



**Figure 13-1. Connection of the Bus Cable**

2. Insert the connector of the bus terminal cable into the PG/OP/SINEC L1 port.

## 13.2 Setting Parameters in the Programmable Controller for Exchanging Data

The programmable controller requires the following information for the handling of data exchange via the L1 bus:

- Location of the data to be sent (data block or flag area)
  Name: **Send Mailbox**, abbreviated: **SF**
- Location of the data to be received (data block or flag area)
  Name: **Receive Mailbox**, abbreviated: **EF**
- Storage location of the coordinating information for sending data (e.g., the message: "Send Mailbox is enabled")
  Name: **Coordination Byte Send**, abbreviated: **KBS**

- Storage location of the coordinating information for receiving data (e.g., the message: "Receiving data can be read")
  Name: **Coordination Byte Receive**, abbreviated: **KBE**
- **Programmer number** (necessary if you want to transmit programmer functions over the SINEC L1 local area network), abbreviated: **PGN**

You can set parameters for the programmable controller for the CPU 102 in the function block, and for the CPU 103 version 8MA03 in the integrated data block (DB1).

### 13.2.1 How to Program in a Function Block, for CPU 102 and Higher

You can program the SINEC L1 local area network by first setting the parameters and then programming the "messages" in the control program (see section 13.3.3)

**Setting Parameters in the Function Block**

You must define the following in the program:

- Your own programmer number for the programmer bus functions
- Your own slave number
- The data or flag areas reserved by the send and receive mailboxes
- The location of the coordination bytes

You program in the function block by calling up one of the two restart organization blocks (OB21 or OB22). You store the corresponding parameters in the system data area of the programmable controller by using the "TNB" block transfer statement. The SINEC L1 parameter block begins at system data word 57.

**Table 13-1. SINEC L1 Parameter Block**

| System Data Word | High Byte | Low Byte | Address in System Data Area |
|---|---|---|---|
| SD57 | Programmer number (1 through 30) | Slave number (1 through 30) | $EA72_H$ $EA73_H$ |
| SD58 | KBE Data identifier | KBE DB or flag byte | $EA74_H$ $EA75_H$ |
| SD59 | KBE Data word | KBS Data identifier | $EA76_H$ $EA77_H$ |
| SD60 | KBS DB or flag byte | KBS Data word | $EA78_H$ $EA79_H$ |
| SD61 | SF Data identifier | SF DB or flag byte | $EA7A_H$ $EA7B_H$ |
| SD62 | SF Data word | EF Data identifier | $EA7C_H$ $EA7D_H$ |
| SD63 | EF DB or flag byte | EF Data word | $EA7E_H$ $EA7F_H$ |

You define the position of the coordination bytes and the starting addresses of the send and receive mailboxes in each case by three bytes.

**Table 13-2. Setting Parameters in the Coordination Byte**

| Meaning | Parameters | Address in System Data Area |
|---|---|---|
| “Flag” data identifier | (“F”) 4D | $EA74_H$ |
| Flag byte | 0 to 127 | $EA75_H$ |
| | | $EA76_H$ |
| “Data word” identifier | (“D”) 44 | $EA77_H$ |
| Data block | 2 to 63 | $EA78_H$ |
| Data word | 0 to 255 | $EA79_H$ |

The data identifier is in ASCII code.

**Overflow**

If data packets longer than 64 bytes are received, the information is **not** written beyond the end of the receive mailbox. There is no overflow message. The end of the receive mailbox is flag byte 127 in the flag area or the last present data word (in the data block).

**Example**:

Setting parameters in the S5-100U as slave 1 in function block 1

Definitions:

- “Receive” coordination byte (KBE) Flag byte FY100
- “Send” coordination byte (KBS) Flag byte FY101
- Send mailbox (SF) Data block DB2 from DW0
- Receive mailbox (EF) Data block DB3 from DW0
- Flag bytes FY 64 to 77 are used as buffer areas.

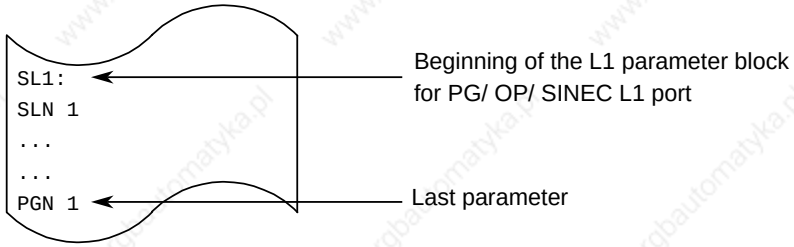
| STL | | | Explanation |
|---|---|---|---|
| L | KF | 1 | - Load slave number and |
| T | FY | 65 | store it in flag byte 65 |
| L | KH | 4D00 | - Load “Flag” data identifier and store it |
| T | FW | 66 | in flag byte 66 |
| L | KY | 100,0 | - Load flag byte 100 and store it |
| T | FW | 67 | in flag byte 67 |
| L | KH | 4D00 | - Load “Flag” data identifier and store it |
| T | FW | 69 | in flag byte 69 |
| L | KY | 101,0 | - Load flag byte 101 and store it |
| T | FW | 70 | in flag byte 70 |
| L | KH | 4400 | - Load “Data word” identifier and store it |
| T | FW | 72 | in flag byte 72 |
| L | KY | 2,0 | - Store DB number “2” and DW number “0” |
| T | FW | 73 | in flag bytes 73 and 74 |
| L | KH | 4400 | - Load “Data word” identifier and store it |
| T | FW | 75 | in flag byte 75 |
| L | KY | 3,0 | - Store DB number “3” and DW “0” |
| T | FW | 76 | in flag bytes 76 and 77 |
| | | | - Transfer flag area FY64 to 67 to the system data area: |
| L | KH | EE4D | - Load upper source address |
| L | KH | EA7F | - Load upper destination address |
| TNB | | 14 | - Transfer a block of data 14 bytes long |
| | | | Erase all buffer areas |
| L | KH | 0000 | - Load hexadecimal number “0000” |
| T | FW | 64 | - Set all bits of FY64 to 77 to “0” |
| T | FW | 66 | |
| T | FW | 68 | |
| T | FW | 70 | |
| T | FW | 72 | |
| T | FW | 74 | |
| T | FW | 76 | |
| | | | Default setting of the KBE; |
| | | | Data can be received from the bus. |
| L | KH | 0080 | - Load binary number 1000 0000 |
| T | FY | 100 | - Set bit 7 to “1” and bits 6 to 0 to “0” |
| | | | Default setting of the KBS; |
| | | | Program has access to the send mailbox |
| L | KH | 0000 | - Load binary number 0000 0000 |
| T | FY | 101 | - Set bits 7 to 0 to “0” |
| BE | | | Block end |

**Figure 13-2. Programming Example for Setting Parameters in FB1**

### 13.2.2 Setting Parameters in DB1, for CPU 103 and Higher

Set the parameters in DB1 as follows:

1. Display the default DB1 on the programmer (Transfer function, source: PC, target: FD (PG)
   - A default DB1 is integrated into the programmable controller's operating system; it contains default parameters for the data exchange via SINEC L1.
2. Look for the SINEC L1 parameter block with the block ID “SL1:” for the PG/OP/SINEC L1 port.
3. Overwrite the comment character (#) with a space.




4. Edit the default parameters according to your requirements. Do **not** change the syntax.

   **Example:**

   The S5-100U participates in the SINEC L1 network as a slave with the slave number 2.

   - Send Mailbox (SF) in DB2 beginning with data word 0
   - Receive Mailbox (EF) in DB2 beginning with data word 10
   - Coordination Byte Send (KBS) is flag byte 0 (MB0)
   - Coordination Byte Receive (KBE) is flag byte 2 (MB2)
   - Programmer bus number (PGN) is 1.

Table 13-3 shows how to change default parameters for the example given above and which parameter settings are permitted.

**Table 13-3. Setting Parameters for the SINEC L1 Interface**

| Default DB1: Block: SINEC L1 to PG/ OP/ SINEC L1 Port | Explanation | Modifications Necessary for the Example | Valid Parameters for CPU 103 and Higher |
|---|---|---|---|
| ... | | | |
| SL1: | Block ID “SINEC L1 to Interface SL1” | no modification necessary | — |
| SLN 1 | Slave number of the PLC; default to 1 | SLN 2 | SLN x<br>(x=1 to 30) |
| SF DB2DW0 | Location of the Send Mailbox; default location: DB2 beginning with DW0 | SF DB2DW0 | SF DBxDWy<br>(x=2 to 255; y=0 to 255) or<br>SF MBz<br>(z=0 to 255) |
| EF DB3DW0 | Location of the Receive Mailbox; default location: DB3 beginning with DW0 | EF DB2DW10 | EF DBxDWy<br>(X=2 to 255; y=0 to 255) or<br>EF MBz<br>(z=0 to 255) |
| KBE MB100 | Location of the Coordination Byte Receive; default location flag byte 100 (MB100) | KBE MB2 | KBE MBx<br>(x=0 to 255)<br>or<br>KBE DByDWx*<br>(y=2 to 255; z=0 to 255) |
| KBS MB101 | Location of the Coordination Byte Send; default location flag byte 101 (MB101) | KBS MB0 | KBS MBx<br>(x=0 to 255)<br>or<br>KBS DByDWz*<br>(y=2 to 255; z=0 to 255) |
| PGN 1 | Programmer bus number (necessary if you want to transmit programmer functions over SINEC L1; default 1) | PGN 1<br>no modification necessary | PGN x<br>(x=1 to 30) |

* The KBE/KBS is in the high-order byte of the given data word.

5. Transfer the changed DB1 to the programmable controller. The default DB1 is overwritten.

If you now go from STOP to RUN or from Power OFF to Power ON (with a battery inserted), the programmable controller accepts the changed parameters and stores them in the system data area.

## 13.3 Coordinating Data Exchange in the Control Program

After you set the parameters, the control program for data exchange has to be created. The control program relies on the coordination information that the operating system makes available in the coordination bytes (see Figure 13-3).

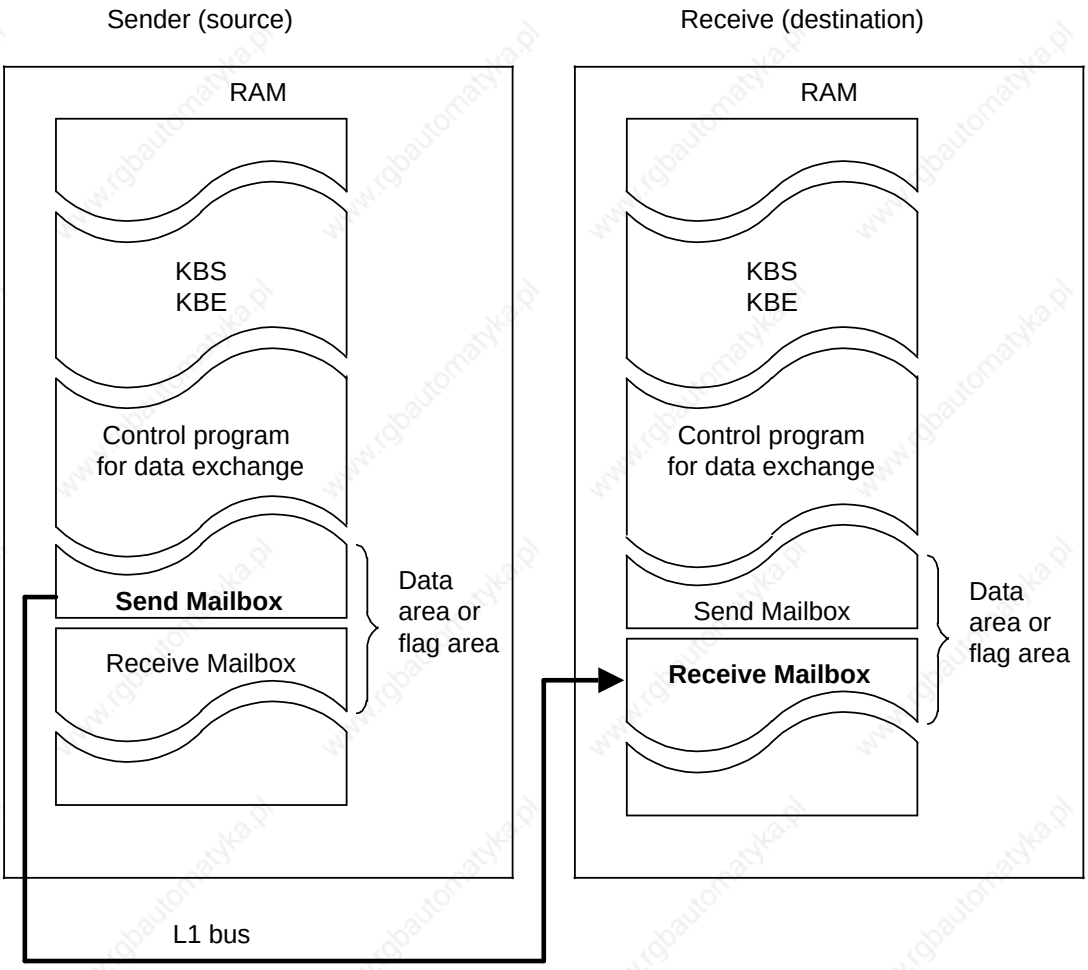



**Figure 13-3. Data Exchange between Sender and Receiver (Principle)**

In the following paragraphs, you will learn how to control the sending and receiving of data after you have set the parameters in DB1.

There is an example in section 13.3.3 of how you must program the data exchange in a function block.

### 13.3.1 Sending Data

The prerequisites for sending data are as follows:

- The parameters are set in DB1 for the location of the Send Mailbox (see section 13.2.2).
- The data to be sent, additional information (length of the send data “net data”), and destination slave number are then transferred to the Send Mailbox.

Figure 13-4 shows which information has to be stored in what part of the Send Mailbox.

**Example:**

Send Mailbox in the **flag area**
(beginning with flag byte 1)

| | |
|---|---|
| Flag byte 1 | Length of the “net data” (0 to 64 bytes) |
| Flag byte 2 | Numb. of the dest. slave* |
| Flag byte 3 | Data (“net data”) maximum of 64 bytes |
| Flag byte 66 | |

**Example:**

Send Mailbox in the **data block**
(beginning with data word DW1)

| | DL | DR |
|---|---|---|
| DW1 | Length of the “net data” | Number of the destination slave* |
| DW2 | 1st data byte | 2nd data byte |
| | | |
| | | |
| DW33 | 63rd data byte | 64th data byte |

* Number of the receiver: 0 = Master
1 to 30 = Slaves
31 = Broadcast

**Figure 13-4. Structure of the Send Mailbox**

**Structure of the Coordination Byte Send (KBS)**

Figure 13-5 shows the structure of the Coordination Byte Send (KBS).

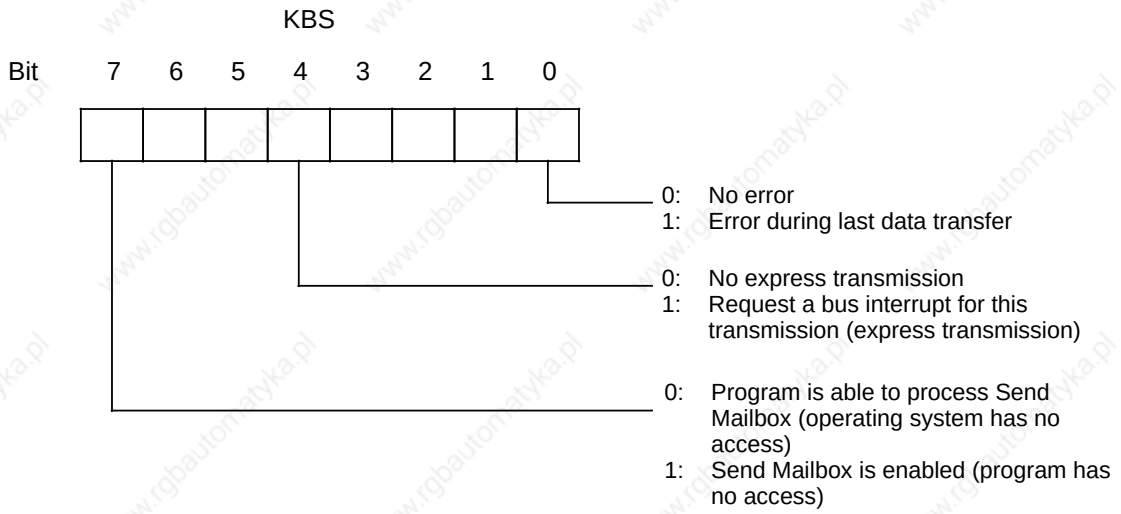



**Figure 13-5. Structure of the Coordination Byte Send (KBS)**

The control program for sending data should be structured as follows:

1. Check bit 7 in the KBS to see if data is currently being sent.
   - If the programmable controller is sending data, bit 7 is set. During this phase, the Send Mailbox can not be modified and no transmission can be started.
2. When bit 7 in the KBS has been reset, you can start the transmission by setting bit 7.
3. When bit 7 has been reset by the operating system after data has been sent, you can evaluate possible errors.

You achieve the following by setting bit 4 in the KBS (express transmission):

- The sending programmable controller treats this message preferentially (possibly by overwriting a telegram not yet sent).
- The receiver treats the message as an express transmission.

In case of an error, the operating system sets bit 0 of the KBS. The error message is not valid until bit 7 has been reset in the KBS.

### 13.3.2 Receiving Data

The prerequisites for receiving data are as follows:
The parameters for the location of the Receive Mailbox and the Coordination Byte Receive (KBE) (see section 13.2.2) have been set in DB1. Figure 13-6 shows you which information has to be stored in what part of the Receive Mailbox.

**Example:**

Receive Mailbox in the **flag area**
(beginning with flag byte 1)

| | |
|---|---|
| Flag Byte 1 | Length of "net data" (in bytes) |
| Flag Byte 2 | Source slave number* |
| Flag Byte 3 | Data ("net data") |

**Example:**

Receive Mailbox in a **data block**
(beginning with data word 1)

| | DL | DR |
|---|---|---|
| DW1 | Length of the "net data" | Source slave number * |
| DW2 | 1st data byte | 2nd data byte |
| DW3 | 3rd data byte | 4th data byte |
| | | |

* Number of the sender: 0 = Master
1 to 30 = Slave

**Figure 13-6. Structure of the Receive Mailbox**

**Structure of the Coordination Byte Receive (KBE)**

Figure 13-7 shows the structure for receiving data (KBE).

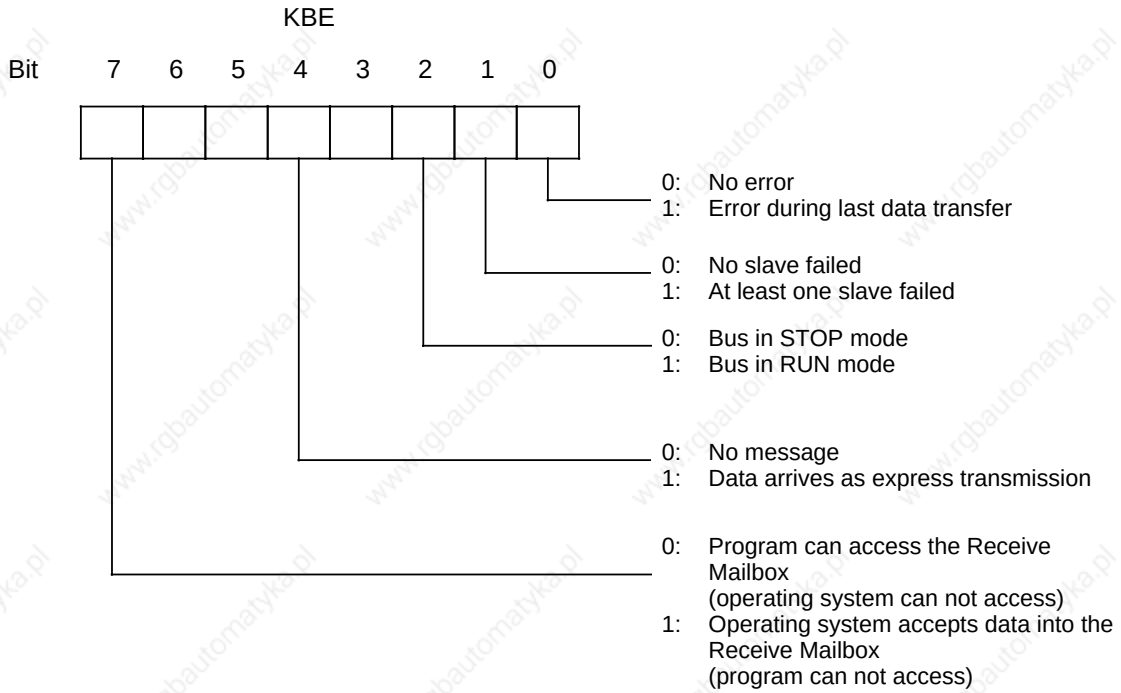



**Figure 13-7. Structure of the Coordination Byte Receive (KBE)**

**Structure of the Control Program for Receiving Data:**

1. Check bit 7 of the KBE to see if it is possible to read the data from the Receive Mailbox. Bit 7 must be set to “0” so that the Receive Mailbox can be read.
2. In addition, you can scan through the KBE for the following errors and operating conditions:
   - At least one slave has failed.
   - The bus is in RUN (STOP) mode.
   - The received data pack comes as an express transmission.

**Special Features**

If you have reserved too little memory for the Receive Mailbox, the available memory area is filled up completely (flag area FY0 to FY255, DW0 to DW255). Therefore, the remaining receive data cannot be stored. In this case, the programmable controller does not generate an overflow message.

You can find sample programs for sending and receiving data in the *SINEC L1* manual (in the chapter on “Programming”).

### 13.3.3 Programming the Messages in a Function Block

The control program must execute the following tasks:

- Enable the send and receive mailboxes and process the data contained in them.
- Manage the coordination bytes (e.g. send request, error evaluation).

**Example:**

Data traffic with the master as slave 1

Definitions:

- Slave 1 receives three bytes from Master 0.
- The information is stored in the process output image table (QB0, QB1, QB2).
- Slave 1 sends three bytes (IB0, IB1, IB2) to the master.
- Parameters are set in FB1 as shown in Figure 13-2.

Programming the individual blocks:

| STL | Explanation |
|---|---|
| **OB22:**<br>JU FB1<br><br>BE | OB22 is processed once only following power up. It calls FB1, which assigns the parameters to the slave. |
| **OB1:**<br>.<br>.<br>JU FB2<br>.<br>.<br>.<br>BE | OB1 is scanned cyclically, and calls FB2, which services the send and receive mailboxes. |

**Figure 13-8. Organization of Program Execution**

| STL | | | Explanation |
|---|---|---|---|
| | C | DB3 | Receive mailbox (DB3) |
| | A | F100.7 | Check whether access to receive mailbox is permissible.<br>KBE/Bit 7=0: Access permitted<br>KBE/Bit 7=1: Access not permitted |
| | JC | =M001 | Skip receive mailbox evaluation if access not permitted |
| | L | DR0 | Check whether the number of the source (master 0) is in |
| | L | KF+0 | byte 2 of the receive mailbox |
| | ><F | | |
| | JC | =M002 | Skip receive mailbox evaluation if source No. 0 |
| | L | DL1 | |
| | T | QB0 | |
| | L | DR1 | Transfer receive mailbox |
| | T | QB1 | to the PIQ |
| | L | DL2 | |
| | T | QB2 | |
| M2: | AN | M100.7 | Set KBE/Bit 7=1, i.e. permit PLC access.<br>Program access is not permitted again until the |
| | S | F100.7 | PLC has reset this bit. |
| M1: | A | F101.7 | Check whether access to the send mailbox is permitted.<br>KBS/Bit 7=0: Access permitted<br>KBS/Bit 7=1: Access not permitted |
| | JC | =M003 | Skip send mailbox evaluation if access not permitted. |
| | C | DB2 | Set send mail box (DB2) |
| | L | KF+3 | Specify length of the data packet in byte 1 |
| | T | DL0 | of the send mailbox |
| | L | KF+0 | Load destination number 0 (master) into byte 2 of the |
| | T | DR0 | send mailbox |
| | L | IB3 | |
| | T | DL1 | Load input bytes 3, 4 and 5 |
| | L | IB4 | into the send mailbox |
| | T | DR1 | |
| | L | IB5 | |
| | T | DL2 | |
| | AN | F101.7 | Set KBS/bit 7, i.e. programmable controller has access to the send |
| | S | F101.7 | mailbox |
| M3: | NOP | 0 | |
| BE | | | |

**Figure 13-9. Programming “Message Processing” in FB2**

## 14 Module Spectrum


| | | | |
|---|---|---|---|
| 14.1 | General Technical Specifications | 14 - 3 |
| 14.2 | Power Supply Modules | 14 - 4 |
| 14.3 | Central Processing Units | 14 - 7 |
| 14.4 | Bus Units | 14 - 10 |
| 14.5 | Interface Modules | 14 - 14 |
| 14.6 | Digital Modules | 14 - 16 |
| 14.6.1 | Digital Input Modules | 14 - 16 |
| 14.6.2 | Digital Output Modules | 14 - 26 |
| 14.6.3 | Digital Input/Output Modules | 14 - 36 |
| 14.7 | Analog Modules | 14 - 38 |
| 14.7.1 | Analog Input Modules | 14 - 38 |
| 14.7.2 | Analog Output Modules | 14 - 56 |

# 14 Module Spectrum

The following section describes the standards and test values the S5-100U meets and fulfills and the test criteria with which the S5-100U has been tested.

**UL/CSA approvals**

The following approvals exist for the S5-100U:
UL Recognition Mark
Underwriters Laboratories (UL) in accordance with Standard UL 508, File E 116536
CSA Certification Mark
Canadian Standard Association (CSA) in accordance with Standard C 22.2 No. 142, File LR 48323

**CE marking**

Our products meet the requirements and protection objectives of the following EC Directives and comply with the harmonized European standards (EN) published in the Official Gazettes of the European Communities with regard to programmable controllers:

- 89/336/EC "Electromagnetic Compatibility" (EMC Directive)
- 73/23/EC "Electrical Equipment Designed for Use between Certain Voltage Limits" (Low-Voltage Directive)

CE The EC declarations of conformity are held at the disposal of the competent authorities at the address below:

Siemens Aktiengesellschaft
Bereich Automatisierungstechnik
AUT E 14
Postfach 1963
D-92209 Amberg
Federal Republic of Germany

**Area of Application**

SIMATIC products have been designed for use in industrial environments.
With indiivdual approval, SIMATIC products can also be used in residential environments (residential, commercial and light industry).
You must acquire the individual approval from the respective national authority or testing board.

| Area of Application | Requirements in respect of | |
|---|---|---|
| | Emitted interference | Immunity |
| Industry | EN 50081-2 : 1993 | EN 50082-2 : 1995 |
| Residential | Individual approval | EN 50082-1 : 1992 |

**Oberserving the Installation Guidelines**

S5 modules meet the requirements, if installed and operated in accordance with the Installation Guidelines (see chapter 3).

**Notes for the machine manufacturer**

The SIMATIC automation system is not a machine in the sense of the EC Directives Machines. Therefore a declaration of conformity with regard to the EC Directive Machines 89/392/EC does not exist for SIMATIC.

The EC Directive Machines 89/392/EC regulates the requirements on a machine. A machine in this sense is a group of interconnected parts or devices (see also EN 292-1, para. 3.1).

The SIMATIC is part of the electrical equipment of a machine and must therefore be included in the declaration of conformity procedure by the machine manufacturer.

The standard EN 60204-1 (safety of machines, general requirements for the electrical equipment of machines) applies for the electrical equipment of machines.

The following table is intended to help you with the declaration of conformity and shows which criteria apply to SIMATIC in accordance with EN 60204-1 (June 1993).

| EN 60204-1 | Topic/criterion | Remarks |
|---|---|---|
| Para. 4 | General requirements | Requirements are fulfilled if the devices are assembled/installed in accordance with the Installation Guidelines.<br>Please also observe the section on CE marking. |
| Para. 11.2 | Digital input/output interfaces | Requirements are fulfilled |
| Para. 12.3 | Programmable equipment | Requirements are fulfilled if the devices are installed in lockable cabinets to protect them from memory modifications through unauthorized persons. |
| Para. 20.4 | Voltage tests | Requirements are fulfilled |

## 14.1 General Technical Specifications

| **Climatic Environmental Conditions** | |
|---|---|
| **Temperature** | |
| Operating | |
| - horizontal design | 0 to+60° C (32 to 140° F) |
| - vertical design | 0 to+40° C (32 to 104° F) (Air-intake temperature, measured on the underside of the module) |
| Nonoperating | - 40 to +70° C (- 40 to +150° F) |
| Temperature change | |
| - operating | max. 10° C/h (50° F/h) |
| - nonoperating | max. 20° C/h (68° F/h) |
| **Relative humidity** | to DIN 40040 15 to 95% (indoors), noncondensing |
| **Atmospheric pressure** | |
| - operating | 860 to 1060 hPa |
| - nonoperating | 660 to 1060 hPa |
| **Pollutants** | |
| - $SO_2$ | 0.5 ppm, (rel. humidity 60%, noncondensing) |
| - $H_2S$ | 0.1 ppm, (rel. humidity 60%, noncondensing) |

| **Electromagnetic Compatibility (EMC) Noise Immunity** | |
|---|---|
| **Static electricity** | to IEC 801-2 |
| (discharge on all parts that are accessible to the operator during normal operation) | |
| - Test voltage | 2.5 kV (relative humidity 30 to 95%) |
| **Radiated electromagnetic field test** | to IEC 801-3 field strength 3 V/m |
| **Fast transient burst** | to IEC 801-4, class III |
| **Power supply modules** | |
| - Supply voltage 24 V DC | 1 kV |
| - Supply voltage 115/230 V AC | 2 kV |
| - Analog input/output modules | 1 kV |
| - Digital input/output modules | |
| for V = 24 V | 1 kV |
| for V > 24 V | 2 kV |
| **Communications interface** | 1 kV |
| **Emitted interference** | to VDE 0871 Limit value class A |

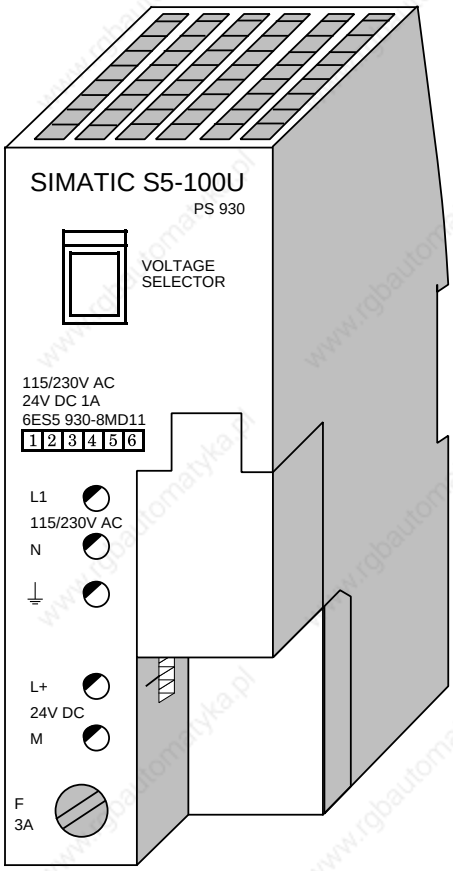
| **Mechanical Environmental Conditions** | |
|---|---|
| **Vibration*** | |
| - tested to IEC 68-2-6 | |
| 10 to 57 Hz, | Const. ampl. 0.075 mm |
| 57 to 150 Hz, | Const. accel. 1 g |
| Mode of vibration | Frequency sweeps with a sweep rate of 1 octave/min |
| Vibration period | 10 frequency sweeps per axis in each of the 3 axes vertical to each other |
| **Shock*** | |
| - tested to IEC 68-2-27 | |
| Type of shock | Half sine |
| Strength of shock | 15 g peak value, 11 ms duration |
| Direction of shock | 2 shocks in each of the 3 axes vertical to each other |
| **Free-fall** | to IEC 68-2-31 |
| - tested with | height of fall 50 mm |

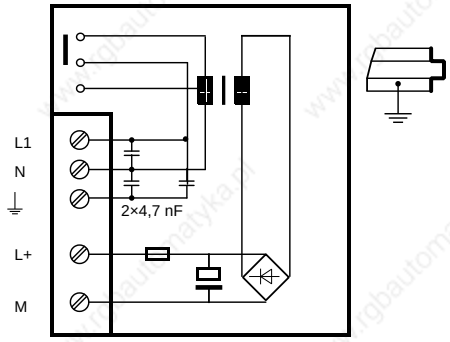
| **IEC/VDE Safety Information** | |
|---|---|
| **Degree of protection** | to IEC 529 |
| - Type | IP 20 |
| - Class | to IEC 536 |
| **Insulation rating** | to VDE 0160 |
| - between electrically independent circuits **and** with circuits connected to a central grounding point | (05. 1988) to VDE 0160 |
| - between all circuits **and** a central grounding point (standard mounting rail) | (05. 1988) to VDE 0160 (05. 1988) |
| Test voltage | Sine, 50 Hz |
| for a rated voltage $V_{input}$ of the circuits (AC/DC) | |
| $V_{input}$= 0 to 50 V | 500 V |
| $V_{input}$= 50 to 125 V | 1250 V |
| $V_{input}$= 125 to 250 V | 1500 V |

***** Appropriate measures must be taken to avoid vibration, shock and repetitive shock.

## 14.2 Power Supply Modules

### Power Supply Module PS 930 115/230 V AC; 24 V DC/1 A (6ES5 930-8MD11)



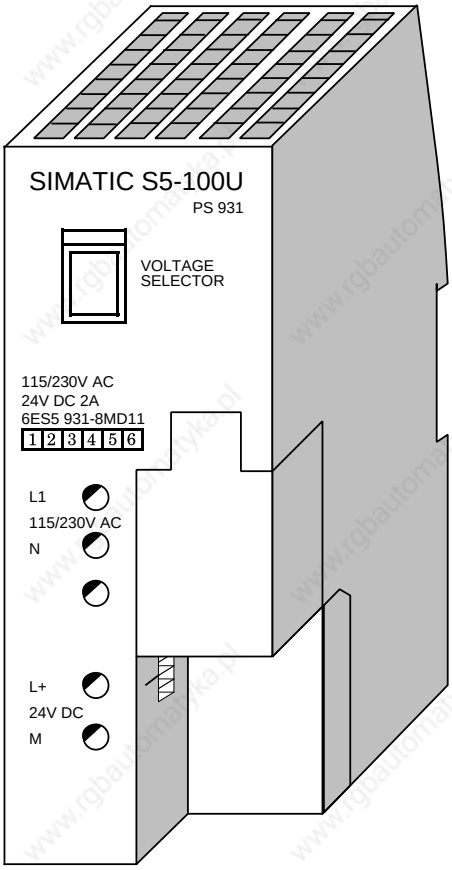

**Technical Specifications**

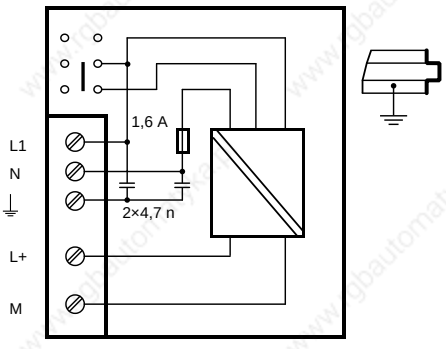
| | | |
|---|---|---|
| Input voltage | | |
| - rated value | | 115/230 V AC |
| - permiss. range | | 92 to 132 V/<br>187 to 264 V |
| Line frequency | | |
| - rated value | | 50/60 Hz |
| - permiss. range | | 47 to 63 Hz |
| Input current at 115/230 V | | |
| - rated value | | 0.35/0.18 A |
| - Input current | max. | 6/3 A |
| Power consumption | | 33 W |
| Output voltage | | |
| - rated value | | 24 V DC |
| - Permiss. range | | 18 to 34 V[1] |
| - Open-circuit voltage | max. | 39 V |
| Output current | | |
| - rated value | | 1 A |
| Short-circuit | | 3A fast fuse |
| Fault LED | | none |
| Protection class | | class 1 |
| Galvanic isolation | | yes |
| Conductor cross sectional area | | |
| - stranded [2] | | 2×0.5 to 1.5 mm$^2$ |
| - solid | | 2×0.5 to 2.5 mm$^2$ |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+24 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 1500 V AC |
| Ri specification | | A to VDE 0871 |
| Dimensions | | |
| W×H×D in mm | | 45.4×135×120 |
| in. | | 1.8 x 5.3 x 4.7 |
| Power loss of the module | typ. | 7.5 W |
| Weight | approx. | 1040 g (2.4 lbs) |

[1] For this reason, can only be used with the S5-100U CPUs

[2] With core end sleeves

## Power Supply Module PS 931 115/230 V AC; 24 V DC/2 A (6ES5 931-8MD11)



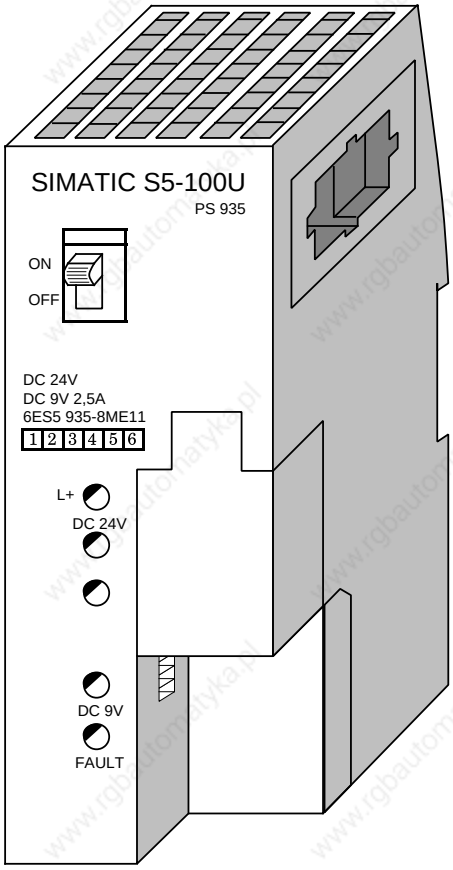



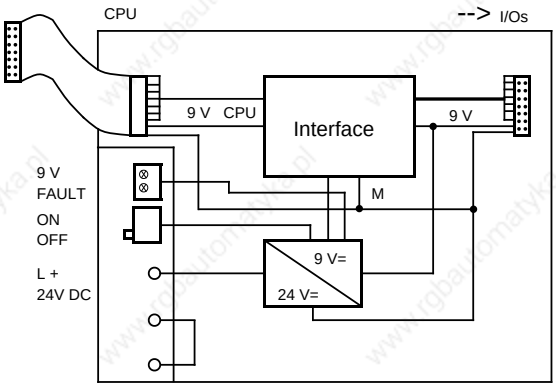

**Technical specifications**

| | | |
|---|---|---|
| Input voltage | | |
| - rated value | | 115/230 V AC |
| - permiss. range | | 92 to 132 V/<br>187 to 264 V |
| Line frequency | | |
| - rated value | | 50/60 Hz |
| - permiss. range | | 47 to 63 Hz |
| Input current at 115/230 V | | |
| - rated value | | 0.9/0.6 A |
| Efficiency | approx. | 85% |
| Power consumption | approx. | 60 W |
| Output voltage | | |
| - rated value | | 24 V DC |
| - permiss. range | | 22.8 to 25.2 V |
| - open-circuit voltage | | yes |
| Output current | | |
| - rated value | | 2 A |
| Permiss. ambient temperature of PLC | | |
| - horizontal arrangement | | 0 to 60 °C<br>(32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C<br>(32 to 104 °F) |
| Buffering of line voltage dips | | |
| - duration of voltage dips | | 20 ms at 187 V/2 A |
| - repetition rate | | 1 s |
| Short-circuit protection | | power limiting, electronic cutoff, non-latching |
| Fault LED | | no |
| Protection class | | class 1 |
| Galvanic isolation | | yes |
| Conductor cross-sectional area | | |
| - stranded * | | 2×0.5 to 1.5 mm$^2$ |
| - solid | | 2×0.5 to 2.5 mm$^2$ |
| Insulation rating | | VDE 0160 and VDE 0805 (transformer) |
| Rated insulation voltage (+24 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 2830 V AC |
| Dimensions W×H×D in mm | | 45.4×135×120 |
| in. | | 1.8 x 5.3 x 4.7 |
| Power loss of the module | typ. | 8.5 W |
| Weight | approx. | 500 g (1.1 lbs.) |

* with core end sleeves

## Power Supply Module PS 935 (not with CPU 100) (6ES5 935-8ME11)



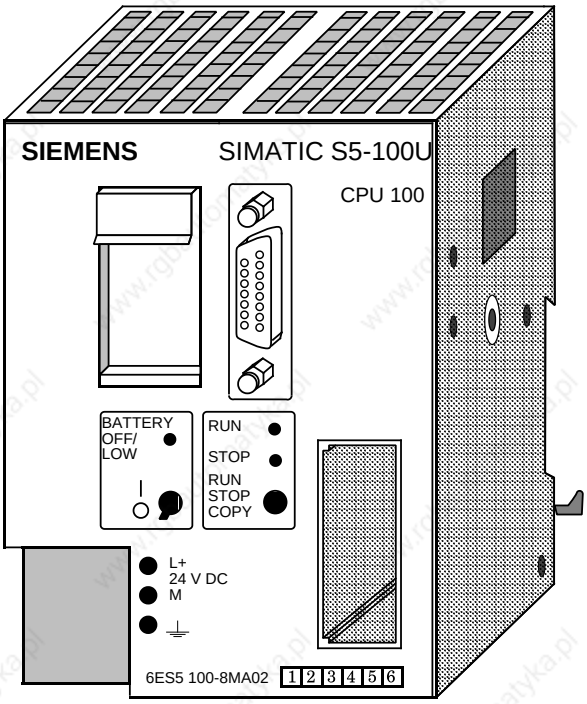

**Technical Specifications**

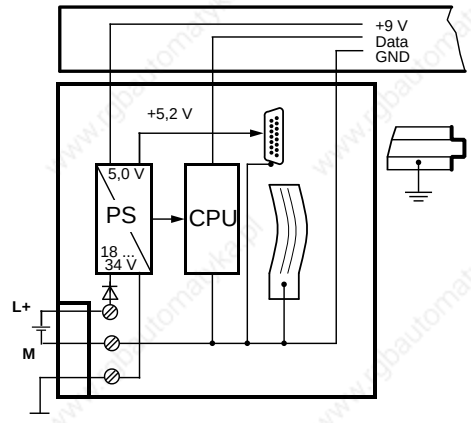
| | | |
|---|---|---|
| Number of inputs (only internal) | | 2×4 bits |
| Input voltage | | |
| - rated value | | 24 V DC |
| - permissible range | dyn. | 18.5 to 30.2 V DC |
| | stat. | 20.4 to 28.8 V DC |
| - Polarity reversal protection | | yes |
| Radio interference level | | A to VDE 0871 |
| Input current at 24 V DC | | |
| - rated value | | 1.25 A |
| - inrush current limitation | | 15 times rated current |
| - efficiency | approx. | 75% |
| Power input | approx. | 30 W |
| Output voltage | | |
| - rated value | | 9 V DC |
| - permissible range | | 8.55 to 9.45 V |
| - open-circuit voltage | | yes |
| Output current | | |
| - rated value | | 2.5 A |
| - permissible range | | 0.0 to 2.5 A |
| - overload recognition | | 2.5 to 2.7 A |
| Buffering during mains voltage dips | | |
| - duration of voltage dips | | 20 ms at 20.4 V/2.5 A |
| - repeat rate | | 1 s |
| Short-circuit protection (output side) | | yes<br>electronic switch-off, non-retentive |
| Diagnostics | | yes (24 V DC input voltage, 9 V output voltage) |
| Fault indication | | yes |
| Class of protection | | Class 1 |
| Galvanic isolation | | no |
| Conductor cross-section | | |
| - stranded* | | 2×0.5 to 1.5 $mm^2$ |
| - solid | | 2×0.5 to 2.5 $mm^2$ |
| Dimensions | | |
| W×H×D mm | | 45.4×135×120 |
| in. | | 1.8×5.3×4.7 |
| Power loss of the module | typ. | 7.5 W |
| Weight | approx. | 500 g (1.1 lbs) |

* with core end sleeve

## 14.3 Central Processing Units (CPUs)

### CPU 100 (6ES5 100-8MA02)



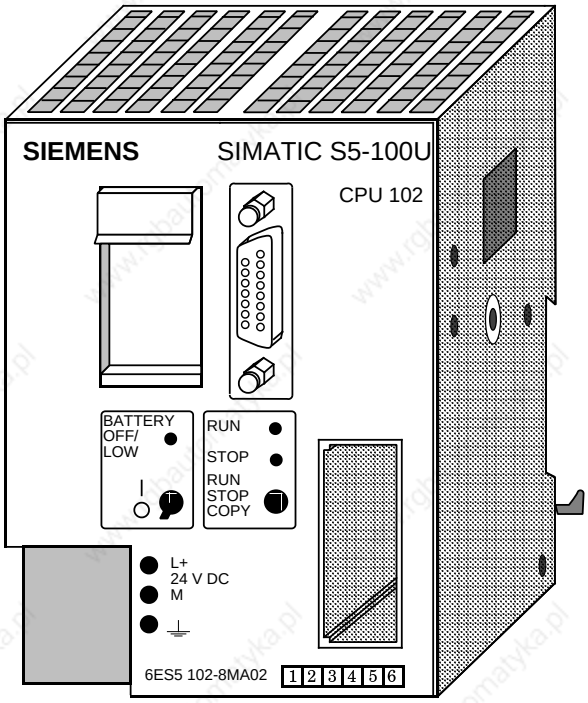

**Technical specifications**

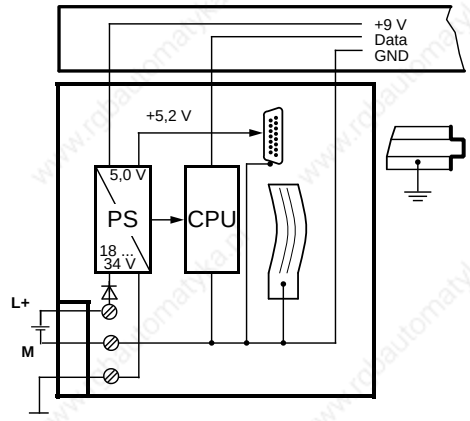
| | | |
|---|---|---|
| Memory configuration | | |
| - internal memory | | RAM 1024 statements |
| - memory submodule | | EPROM/EEPROM |
| Execution times | | |
| - per binary operation | approx. | 70 µs |
| - per word operation | approx. | 125 µs |
| Scan monitoring time | approx. | 300 ms |
| Flags | | 1024; 512 retentive |
| Timers: Number/range | approx. | 16; 0.01 to 9990 s |
| Counters: Number/range | | 16; 8 retentive<br>0 to 999 (up/down) |
| Digital inputs,<br>Digital outputs | together max. | 256 |
| Analog inputs,<br>Analog outputs | together max. | 8 |
| Organization blocks | | OB1, 21, 22, and 34 |
| Program blocks | | 0 to 63 |
| Function blocks | | |
| - programmable | | 0 to 63 |
| - integrated | | none |
| Sequence blocks | | none |
| Data blocks | | 2 to 63 |
| Number of operations | approx. | 60 |

**Power supply (internal)**

| | | |
|---|---|---|
| Input voltage | | |
| - nominal value | | 24 V DC |
| - permiss. range | | 18 to 34 V |
| Current consumption from +24 V | | 1 A |
| Output voltage | | |
| - V 1 (for I/Os) | | +9 V |
| - V 2 (for programmer) | | +5.2 V |
| Output current | | |
| - from V 1 | | 1 A |
| - from V 2 | | 0.65 A |
| Short-circuit protection | | electronic |
| Protection class | | class 1 |
| Galvanic isolation | | no |
| Backup battery | | Lithium Battery (3.4 V/ 850 mAh) |
| - Backup time | min. | 1 year (at 25 °C [77 °F] and uninterrupted backup of CPU) |
| - Service life | approx. | 5 years (at 25 °C [77 °F]) |
| Permissible ambient temperature | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connector cross-sectional area | | |
| - stranded, with core end sleeves | | 2×0.5 to 1.5 $mm^2$ |
| - solid | | 2×0.5 to 1.5 $mm^2$ |
| Power losses of the module | typ. | 10.7 W |
| Dimensions (WxHxD) in | mm | 91.5×135×120 |
| | in. | (3.6 x 5.3 x 4.7) |
| Weight | | |
| - CPU module | approx. | 0.65 kg (1.4 lbs) |
| - memory submodule | approx. | 0.1 kg (0.2 lbs) |

## CPU 102 (6ES5 102-8MA02)

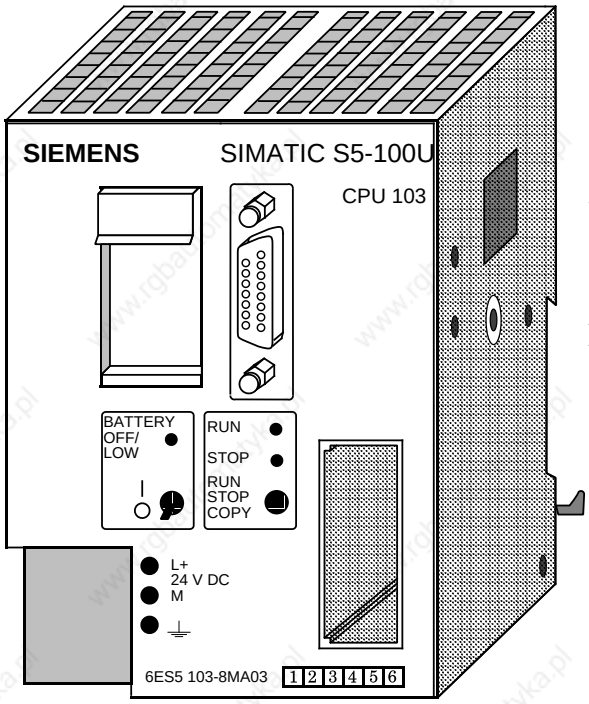



**Technical specifications**

| | | |
|---|---|---|
| Memory configuration | | |
| - internal memory | | RAM 2048 statements |
| - memory submodule | | EPROM/EEPROM |
| Execution times Normal/Test | | |
| - per binary operation | approx. | 7/70 µs |
| - per word operation | approx. | 40/125 µs |
| Scan monitoring time | approx. | 350 ms |
| Flags | | 1024; 512 retentive |
| Timers: Number/range | approx. | 32; 0.01 to 9990 s |
| Counters: Number/range | | 32; 8 retentive<br>0 to 999 (up/down) |
| Digital inputs,<br>Digital outputs | together max. | 256 |
| Analog inputs,<br>Analog outputs | together max. | 16 |
| Organization blocks | | OB1, 21, 22, and 34 |
| Program blocks | | 0 to 63 |
| Function blocks | | |
| - programmable | | 0 to 63 |
| - integrated | | 240 to 243, 250, and 251 |
| Sequence blocks | | none |
| Data blocks | | 2 to 63 |
| Number of operations | approx. | 60 |
| **Power supply (internal)** | | |
| Input voltage | | |
| - nominal value | | 24 V DC |
| - permiss. range | | 18 to 34 V |
| Current consumption from +24 V | | 1 A |
| Output voltage | | |
| - V 1 (for I/Os) | | +9 V |
| - V 2 (for programmer) | | +5.2 V |
| Output current | | |
| - from V 1 | | 1 A |
| - from V 2 | | 0.65 A |
| Short-circuit protection | | electronic |
| Protection class | | class 1 |
| Galvanic isolation | | no |
| Backup battery | | Lithium Battery (3.4 V/ 850 mAh) |
| - Backup time | min. | 1 year (at 25 °C [77 °F] and uninterrupted backup of CPU) |
| - Service life | approx. | 5 years (at 25 °C [77 °F]) |
| Permissible ambient temperature of PC | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connector cross-sectional area | | |
| - stranded, with core end sleeves | | 2×0.5 to 1.5 mm² |
| - solid | | 2×0.5 to 2.5 mm² |
| Power losses of the module | typ. | 11.4 W |
| Dimensions (WxHxD) in | mm | 91.5×135×120 |
| | in. | (3.6 x 5.3 x 4.7) |
| Weight | | |
| - CPU module | approx. | 0.65 kg (1.4 lbs) |
| - memory submodule | approx. | 0.1 kg (0.2 lbs) |

## CPU 103 (6ES5 103-8MA03)

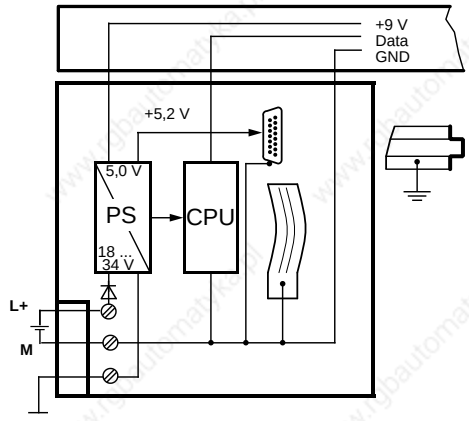



**Technical specifications**

| | | |
|---|---|---|
| Processor | | Byte/bit processor |
| Memory configuration | | |
| - internal memory | | RAM 10240 statements |
| - memory submodule | | EPROM/EEPROM |
| Real-time clock | | |
| - Accuracy | | ±2s/day |
| - Variation due to temperature changes ($T_A$ ambient temperature in °C) | | $-3.5x(T_A-15)^2$ms/day |
| - e.g. tolerance at 40 °C | | ±2s-$3.5x(40-15)^2$ms/day |
| | approx. | 0 to -4s/day |
| Execution times | | |
| - per binary operation | approx. | 0.8 µs |
| - per word operation | approx. | 100 µs |
| Scan monitoring time | | 500 ms, selectable |
| Flags | | 2048; 512 retentive |
| Timers: Number/range | approx. | 128; 0.01 to 9990 s |
| Counters: Number/range | | 128; 8 retentive<br>0 to 999 (up/down) |
| Digital inputs, Digital outputs | together max. | 256 |
| Analog inputs, Analog outputs | together max. | 32 |
| Organization blocks | | OB1, 2, 13, 21, 22, 31, 34, and 251 |
| Program blocks | | 0 to 255 |
| Function blocks | | |
| - programmable | | 0 to 255 |
| - integrated | | 240 to 243, 250 and 251 |
| Sequence blocks | | 0 to 255 |
| Data blocks | | 2 to 255 |
| Number of operations | approx. | 180 |

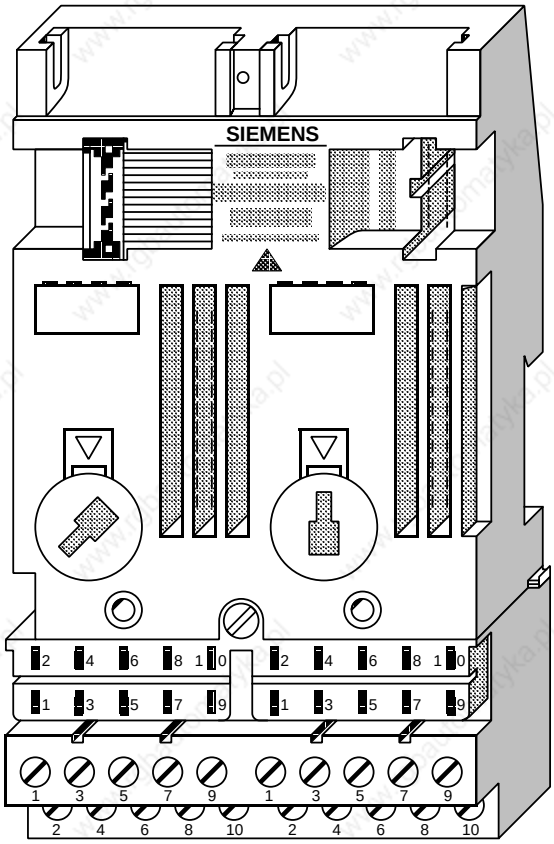
**Power supply (internal)**

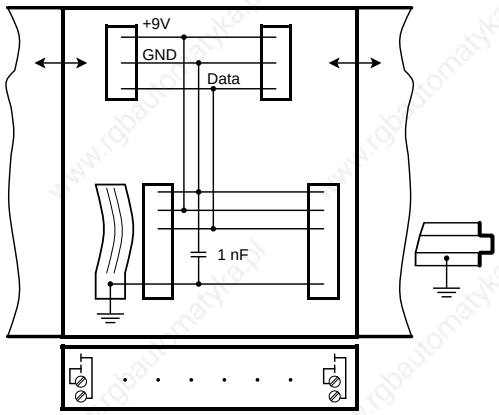
| | | |
|---|---|---|
| Input voltage | | |
| - nominal value | | 24 V DC |
| - permiss. range | | 18 to 34 V |
| Current consumption from +24 V | | 1 A |
| Output voltage | | |
| - V 1 (for I/Os) | | +9 V |
| - V 2 (for programmer) | | +5.2 V |
| Output current | | |
| - from V 1 | | 1 A |
| - from V 2 | | 0.65 A |
| Short-circuit protection | | electronic |
| Protection class | | class 1 |
| Galvanic isolation | | no |
| Backup battery | | Lithium Battery (3.4 V/ 850 mAh) |
| - backup time | min. | 1 year (at 25 °C [77 °F] and uninterrupted backup of CPU) |
| - service life | approx. | 5 years (at 25 °C [77 °F]) |
| Permissible ambient temperature | | |
| - horizontal arrangement | | 0 to 55 °C (32 to 131 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connector cross-sectional area | | |
| - stranded, with core end sleeves | | 2×0.5 to 1.5 mm² |
| - solid | | 2×0.5 to 2.5 mm² |
| Power losses of the module | typ. | 11.6 W |
| Dimensions (WxHxD) in | mm | 91.5×135×120 |
| | in. | (3.6 x 5.3 x 4.7) |
| Weight | | |
| - CPU module | approx. | 0.65 kg (1.4 lbs) |
| - memory submodule | approx. | 0.1 kg (0.2 lbs) |



## 14.4 Bus Units

### Bus Unit (SIGUT Screw-type Terminals) (6ES5 700-8MA11)



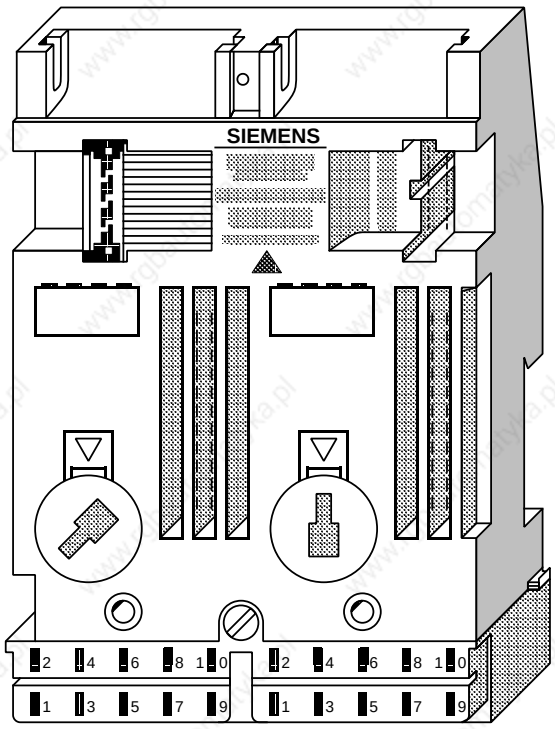



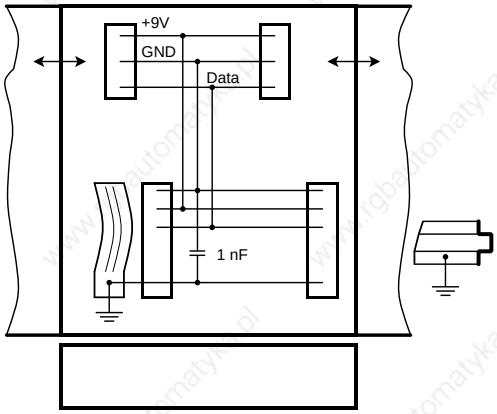

**Technical specifications**

| | | |
|---|---|---|
| Type of connection | | SIGUT screw-type terminals |
| Number of plug-in modules | | 2 |
| Number of bus units per programmable controller | max. | 16 |
| Connection between two bus units | | flat ribbon |
| Number of terminals | | 10 per slot |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Conductor cross sectional area | | |
| - stranded * | | 2×0.5 to 1.5 mm² |
| - solid | | 2×0.5 to 2.5 mm² |
| Current consumption - from +9 V (CPU) | typ. | 1 mA |
| Dimensions W×H×D in mm | | 91.5×162×39 |
| in. | | 3.6×6.4×1.5 |
| Weight | approx. | 300 g (10.6 oz.) |

* with core end sleeves

## Bus Unit (Crimp Snap-in Connections) (6ES5 700-8MA22)

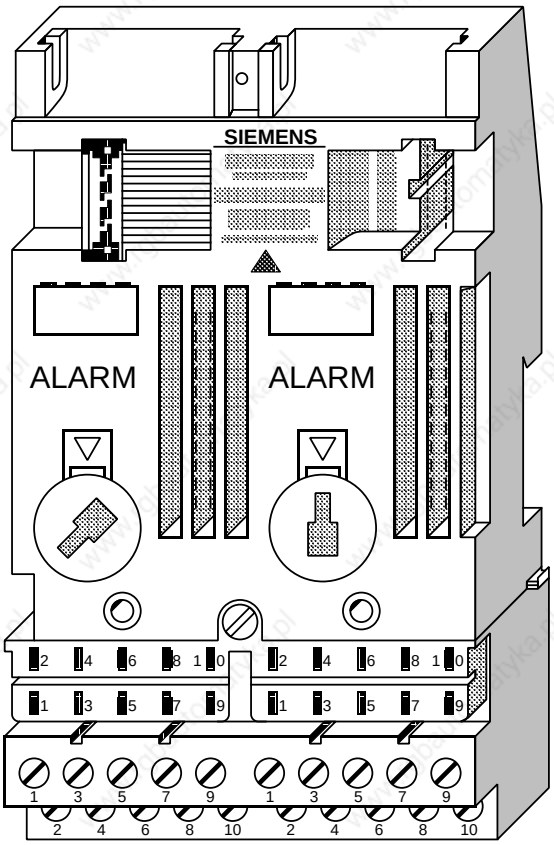


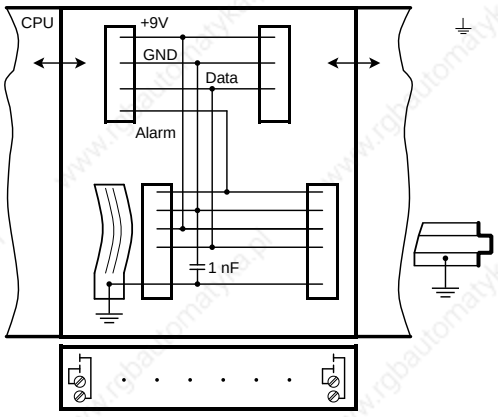




**Technical specifications**

| | | |
|---|---|---|
| Type of connection | | Crimp snap-in |
| Number of plug-in modules | | 2 |
| Number of bus units per programmable controller | max. | 16 |
| Connection between two bus units | | flat ribbon |
| Number of terminals | | 10 per slot |
| Conductor cross sectional area - stranded | | 0.5 to 1.5 mm$^2$ |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption - from +9 V (CPU) | typ. | 1 mA |
| Dimensions W×H×D in mm | | 91.5×135×39 |
| in. | | 3.6×5.3×1.5 |
| Weight | approx. | 250 g (8.8 oz.) |

## Bus Unit with Interrupt Capability (SIGUT Screw-type Terminals) (6ES5 700-8MB11)



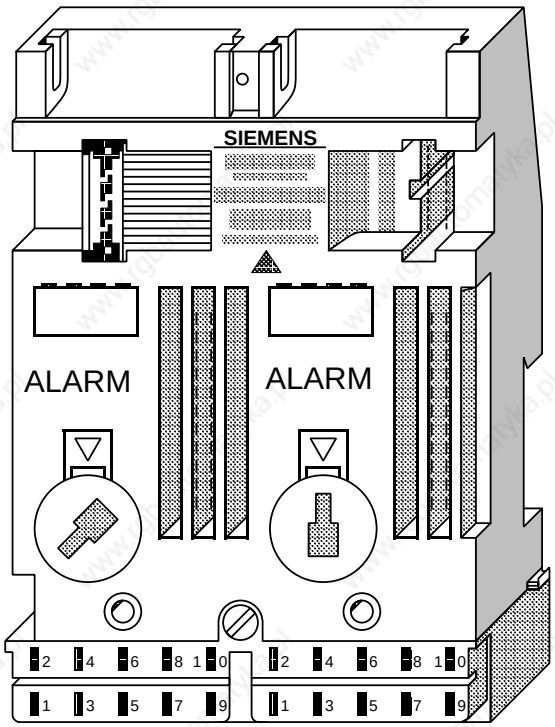

**Technical specifications**

| | | |
|---|---|---|
| Type of connection | | SIGUT (screw-type terminals) |
| Number of plug-in units | | 2 |
| Number of bus modules per programmable controller | max. | 16 * |
| Connection between two bus modules | | flat ribbon |
| Number of terminals | | 10 |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - Insulation group | | 1×B |
| - tested with | | 500 V AC |
| Conductor cross-sectional area | | |
| - stranded ** | | 2×0.5 to 1.5 mm² |
| - solid | | 2×0.5 to 2.5 mm² |
| Current consumption - from+9 V (CPU) | typ. | 11 mA |
| Dimensions (WxHxD) in mm | | 91.5×162×39 |
| in. | | 3.6 x 6.4 x 1.5 |
| Weight | approx. | 320 g (9.8 oz) |

* Only the bus unit (and only with 4-channel digital input modules or comparator modules) directly adjacent to the CPU has interrupt capability

** With core end sleeves

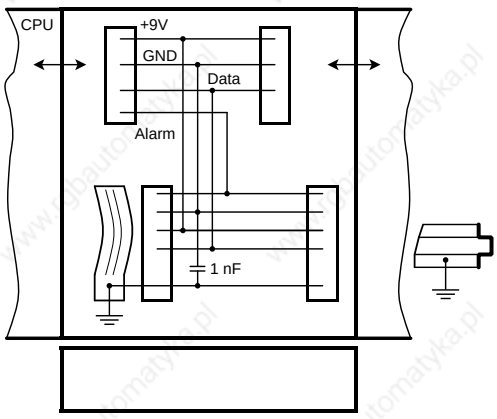
## Bus Unit with Interrupt Capability (Crimp Snap-in Connections) (6ES5 700-8MB21)




**Technical specifications**

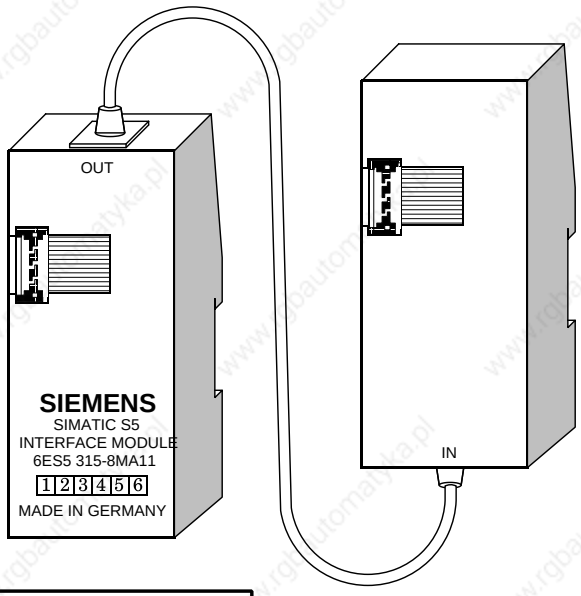
| | | |
|---|---|---|
| Type of connection | | Crimp-snap-in |
| Number of plug-in units | | 2 |
| Number of bus modules per programmable controller | max. | 16 * |
| Connection between two bus modules | | flat ribbon |
| Number of terminals | | 10 per slot |
| Conductor cross-sectional area - stranded | | 0.5 to 1.5 $mm^2$ |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption - from+9 V (CPU) | typ. | 11 mA |
| Dimensions (WxHxD) in mm | | 91.5×135×39 |
| in. | | 3.6 x 5.3 x 1.5 |
| Weight | approx. | 270 g (9.5 oz) |

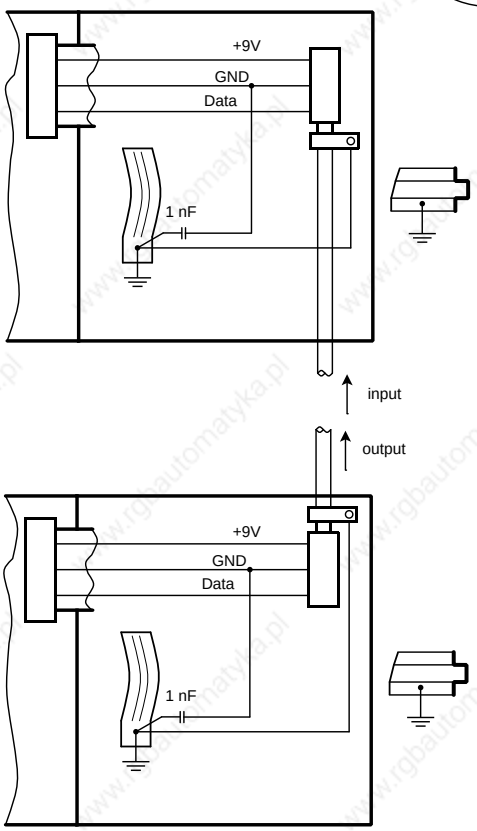
* Only the bus unit (and only with 4-channel digital input modules or comparator modules) directly adjacent to the CPU 103 has interrupt capability



## 14.5 Interface Modules

### IM 315 Interface Module (6ES5 315-8MA11)

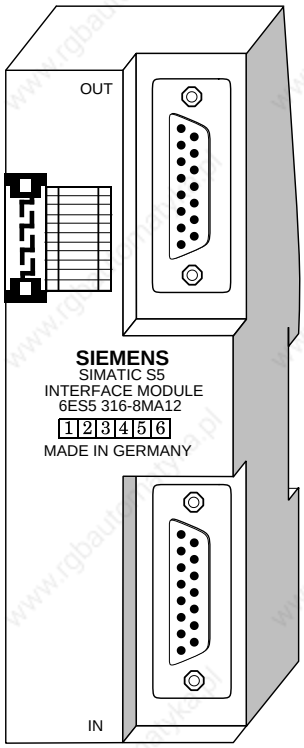


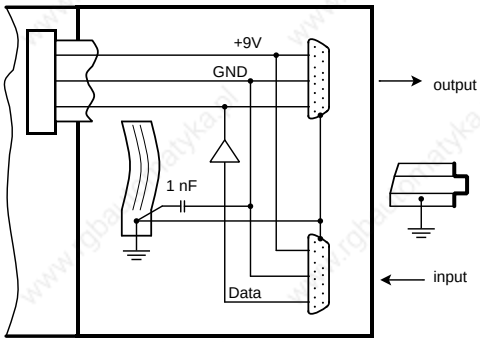




**Technical specifications**

| | | |
|---|---|---|
| Current supply to the expansion unit | max. | 2.5 A |
| Number of interface modules per PLC | max. | 1 |
| Permissible potential difference between ⏚ (IM 315) and central ground point (CPU) | | ±1 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Dimensions W×H×D in mm | | 2×(45.4×135×39) |
| in. | | 2×(1.8×5.3×1.5) |
| Current consumption - from +9 V (CPU) | typ. | 1 mA |
| Weight | approx. | 280 g (9.8 oz.) |

## IM 316 Interface Module (6ES5 316-8MA12)



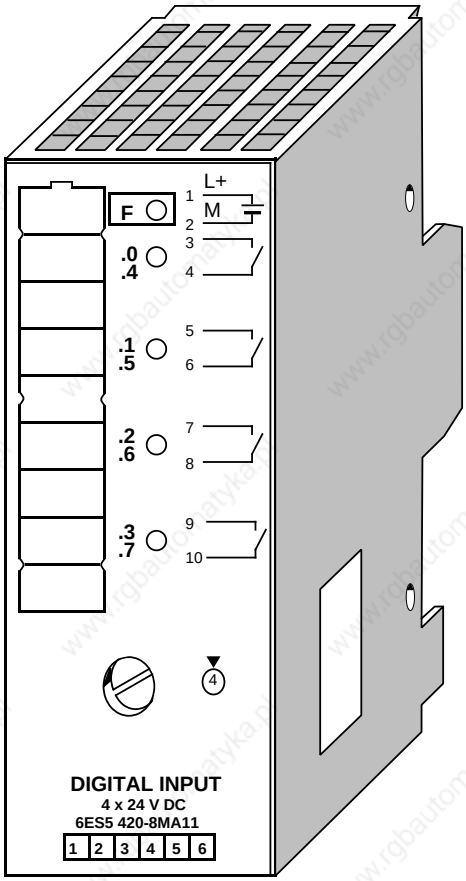

**Technical specifications**

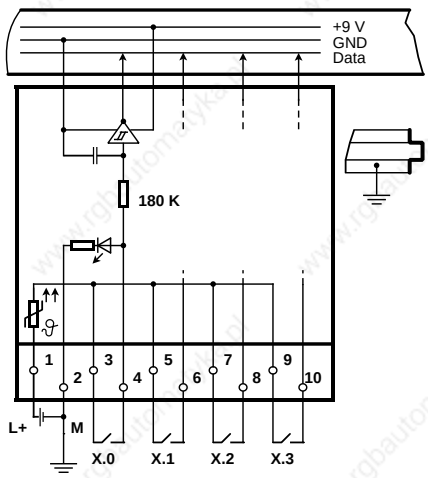
| | | |
|---|---|---|
| Current supply to the expansion unit | max. | 2.5 A |
| Number of interface modules per PLC | max. | 4 |
| Cable connectors for the IM 316 | | |
| - Cable connector (0.5 m/1.6 ft.) | | 6ES5 712-8AF00 |
| - Cable connector (2.5 m/8.2 ft.) | | 6ES5 712-8BC50 |
| - Cable connector (5.0 m/16.4 ft.) | | 6ES5 712-8BF00 |
| - Cable connector (10 m/33 ft.) | | 6ES5 712-8CB00 |
| Cable insulation in ducts | | permissible |
| Permissible potential difference between ⏚ (IM 316) and central ground point (CPU) | | ±1 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| Dimensions W×H×D in mm | | 45.4×135×39 |
| in. | | 1.8×5.3×1.5 |
| Current consumption - from +9 V (CPU) | typ. | 27 mA |
| Weight | approx. | 120 g (4.2 oz.) |

## 14.6 Digital Modules

### 14.6.1 Digital Input Modules

**Digital Input Module 4 × 24 V DC** **(6ES5 420-8MA11)**



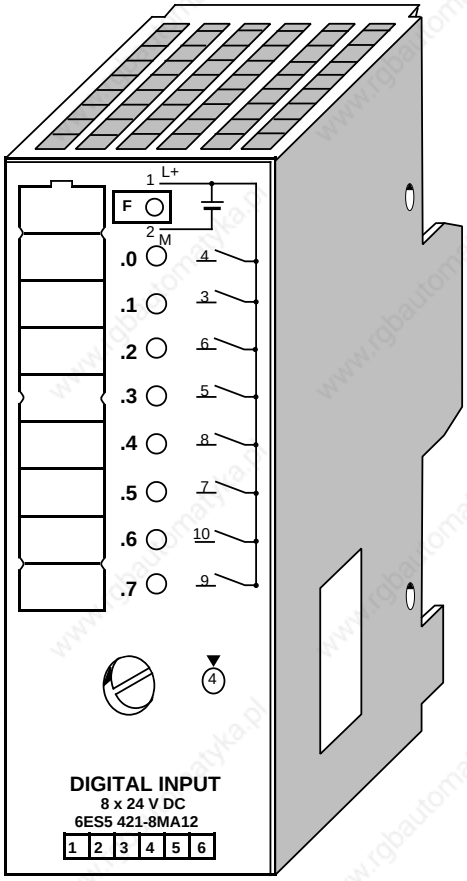



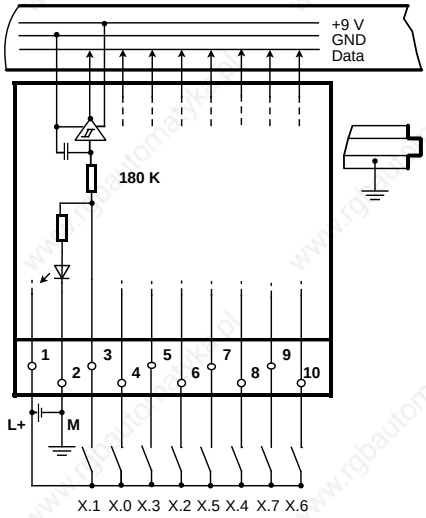

**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 4 |
| Galvanic isolation | | no |
| - in groups of | | 4 |
| Input voltage L+ | | |
| - rated value | | 24 V DC |
| - "0" signal | | 0 to 5 V |
| - "1" signal | | 13 to 33 V |
| Input current at "1" signal | typ. | 7 mA |
| Inherent delay | | |
| - from "0" to "1" | typ. | 2.5 ms |
| - from "1" to "0" | typ. | 5 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage * (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| Fault LED (red) | | no input voltage L+ |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 1.5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 16 mA |
| Power loss of the module | typ. | 0.8 W |
| Weight | approx. | 205 g (7.2 oz.) |

* Relevant only for isolated assembly in the ET 100/200U

## Digital Input Module 8 × 24 V DC (6ES5 421-8MA12)



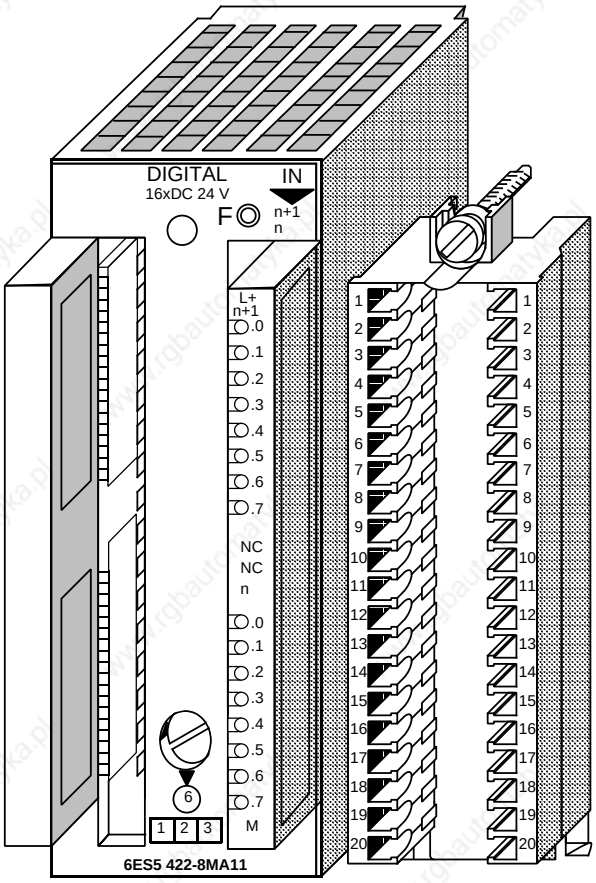

**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 8 |
| Galvanic isolation | | no |
| - in groups of | | 8 |
| Input voltage L+ | | |
| - rated value | | 24 V DC |
| - "0" signal | | 0 to 5 V |
| - "1" signal | | 13 to 33 V |
| Input current at "1" signal | typ. | 7 mA at 24 V |
| Inherent delay | | |
| - from "0" to "1" | typ. | 2.3 ms |
| - from "1" to "0" | typ. | 3.5 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE |
| Rated insulation voltage * (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| Fault LED (red) | | no input voltage L+/M |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 1.5 mA |
| Current consumption from +9 V (CPU) | typ. | 34 mA |
| Power loss of the module | typ. | 1.6 W |
| Weight | approx. | 190 g (6.7 oz.) |

* Relevant only for isolated assembly in the ET 100/200U

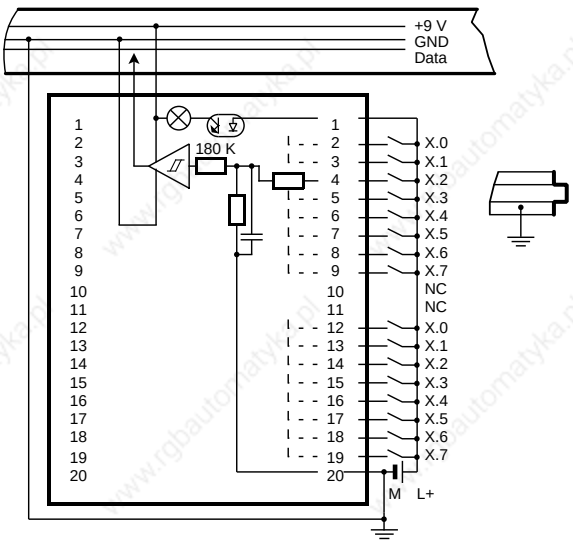
## Digital Input Module 16 × 24 V DC

**(6ES5 422-8MA11)**
**(6ES5 490-8MA13/-8MA03)**
**(6ES5 490-8MB11)**

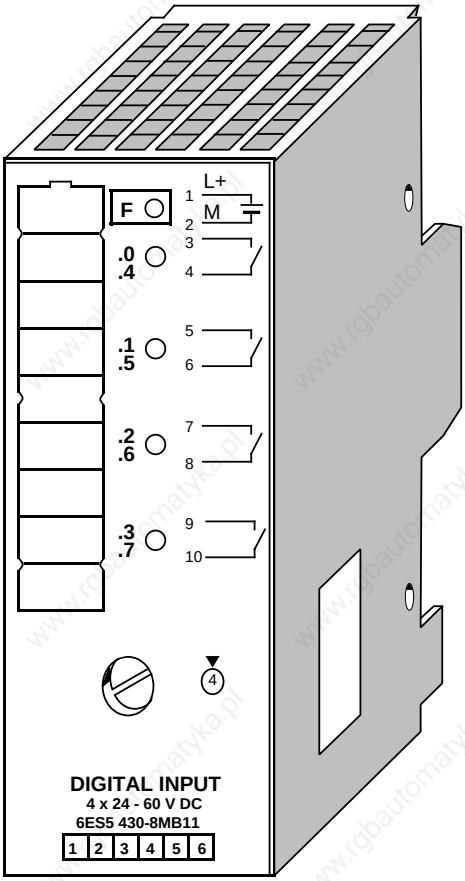


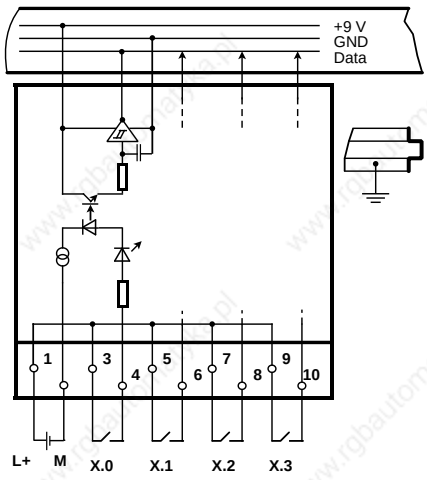

**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 16 |
| Galvanic isolation | | no |
| Input voltage L+ | | |
| - rated value | | 24 V DC |
| - ”0” signal | | 0 to 5 V |
| - ”1” signal | | 13 to 30 V |
| Input protection | | |
| - against polarity reversal | | no, fuse trips |
| - against overvoltage | up to | 33 V |
| Input current at | | |
| ”1” signal | typ. | 4.5 mA |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 4 ms |
| - from ”1” to ”0” | typ. | 3 ms |
| Length of cable | | |
| - unshielded | | 100 m |
| Rated insulation voltage | | |
| (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| EMC/noise immunity to | | |
| VDE 801-4, severity level 3 | | 2 kV |
| Fault LED | | |
| (red) | | on L+/M interruption |
| Connection of 2-wire BERO | | |
| proximity switches | | possible |
| - residual current | | 1.5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 50 mA |
| Power loss of | | |
| the module | typ. | 4.5 W |
| Weight | ca. | 190 g (6.7 oz.) |



## Digital Input Module 4 × 24 to 60 V DC (6ES5 430-8MB11)

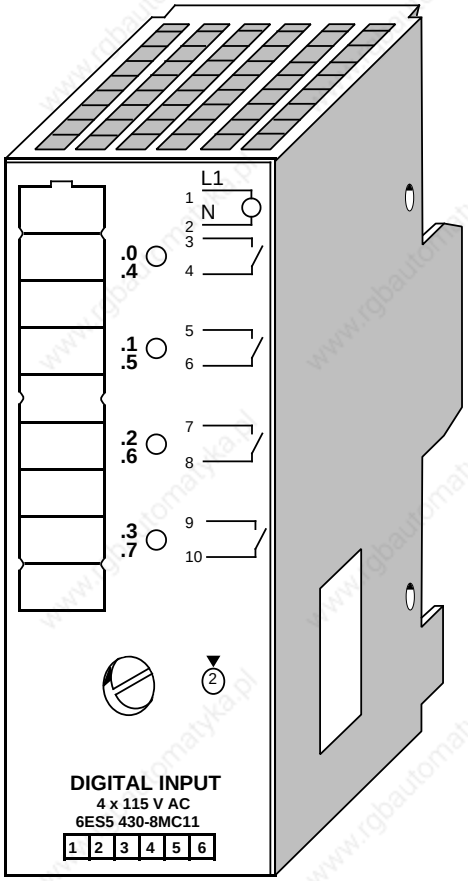


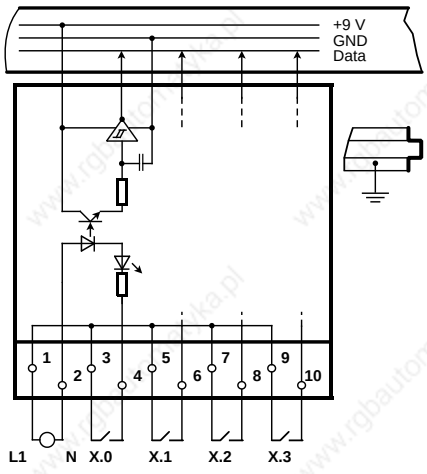




**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 4 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 4 |
| Input voltage L+ | | |
| - rated value | | 24 to 60 V DC |
| - ”1” signal | | 13 to 72 V |
| - ”0” signal | | - 33 to 8 V |
| Input current at ”1” signal | typ. | 4.5 to 7.5 mA |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 3 ms (1.4 to 5 ms) |
| - from ”1” to ”0” | typ. | 3 ms (1.4 to 5 ms) |
| Fault LED (red) | | no input voltage L+ |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 1.5 mA |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+9 V to L+) | | 60 V AC |
| - insulation group | | 2×B |
| - tested with | | 1250 V AC |
| Current consumption | | |
| - from +9 V (CPU) | | 5 mA |
| - from L+ | max. | 35 mA |
| Power loss of the module | max. | 2 W |
| Weight | approx. | 200 g (7 oz.) |

## Digital Input Module 4 × 115 V AC (6ES5 430-8MC11)

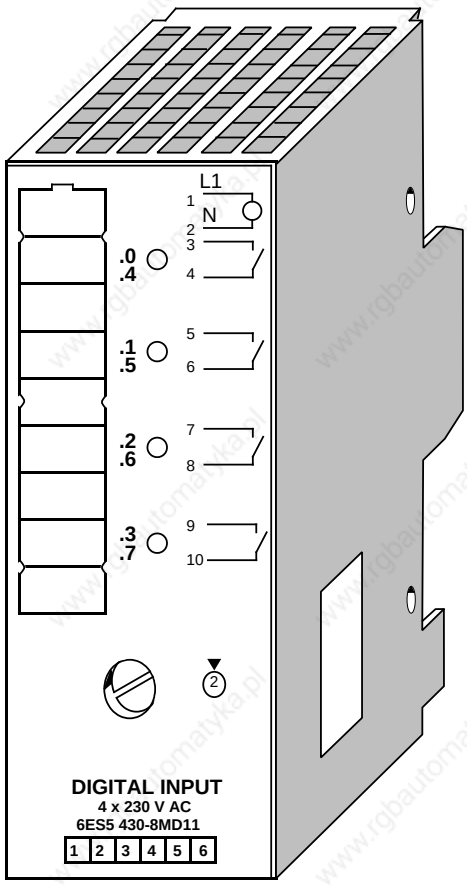


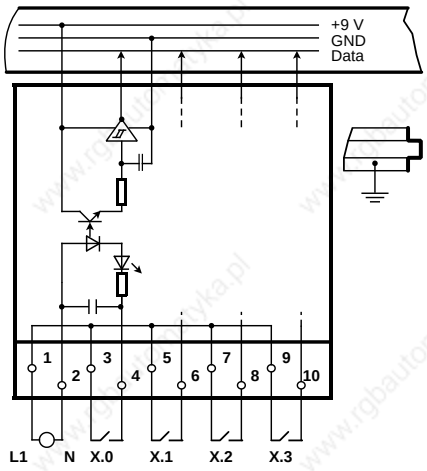




**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 4 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 4 |
| Input voltage L1 | | |
| - rated value | | 115 V AC/DC |
| - ”0” signal | | 0 to 40 V |
| - ”1” signal | | 85 to 135 V |
| - frequency | | 47 to 63 Hz |
| Input current at ”1” signal | typ. | 14 mA at 115 V AC |
| | typ. | 6 mA at 115 V DC |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 10 ms |
| - from ”1” to ”0” | typ. | 20 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 125 V AC |
| - insulation group | | 2×B |
| - tested with | | 1250 V AC |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 16 mA |
| Power loss of the module | typ. | 2.8 W |
| Weight | approx. | 210 g (7.4 oz.) |

## Digital Input Module 4 × 230 V AC (6ES5 430-8MD11)

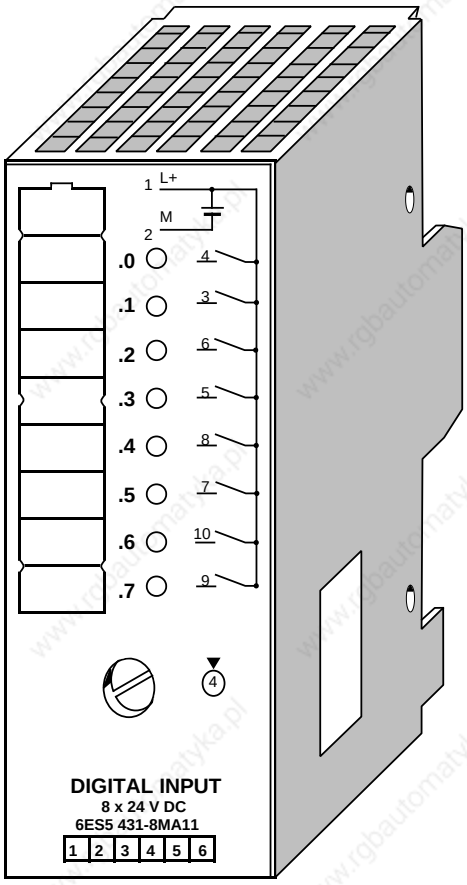


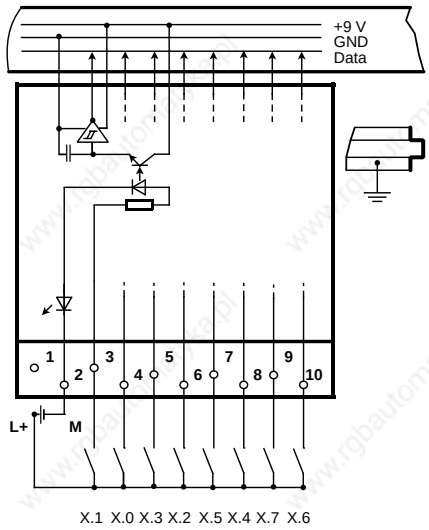




**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 4 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 4 |
| Input voltage L1 | | |
| - rated value | | 230 V AC |
| - ”0” signal | | 0 to 70 V |
| - ”1” signal | | 170 to 264 V |
| - frequency | | 47 to 63 Hz |
| Input current at ”1” signal | typ. | 16 mA at 230 V |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 10 ms |
| - from ”1” to ”0” | typ. | 20 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 1500 V AC |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 16 mA |
| Power loss of the module | typ. | 2.5 W |
| Weight | approx. | 210 g (7.4 oz.) |

## Digital Input Module 8 x 24 V DC (6ES5 431-8MA11)

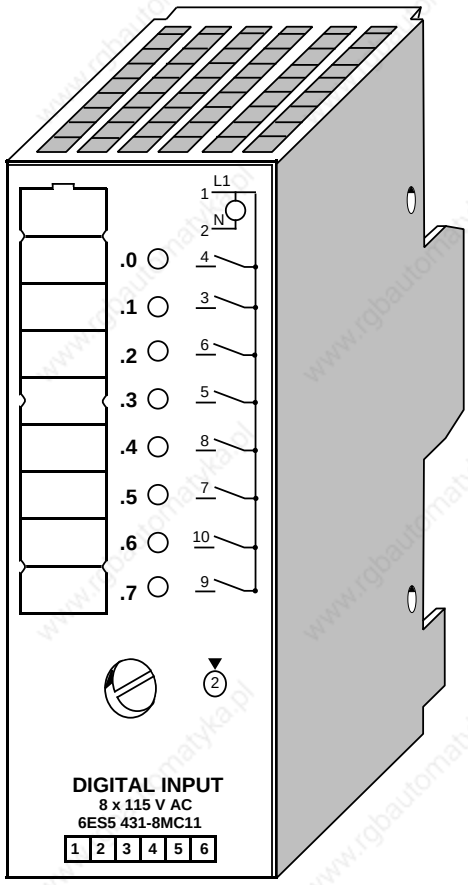


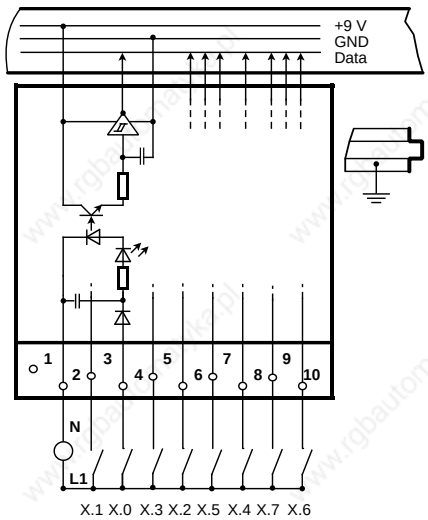




**Technical Specifications**

| | | |
|---|---|---|
| Number of inputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Input voltage L+ | | |
| - rated value | | 24 V DC |
| - "0" signal | | 0 to 5 V |
| - "1" signal | | 13 to 33 V |
| Input current at "1" signal | typ. | 8.7 mA |
| Inherent delay | | |
| - from "0" to "1" | typ | 5.5 ms |
| - from "1" to "0" | typ. | 4.5 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+ 9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 2 x B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+ 9 V to L+) | | 30 V AC |
| - insulation group | | 2 x B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 1.5 mA |
| Current consumption | | |
| - from + 9 V (CPU) | typ. | 32 mA |
| Power loss of the module | typ. | 2 W |
| Weight | approx. | 190 g (6.7 oz.) |

## Digital Input Module 8 × 115 V ACV (6ES5 431-8MC11)

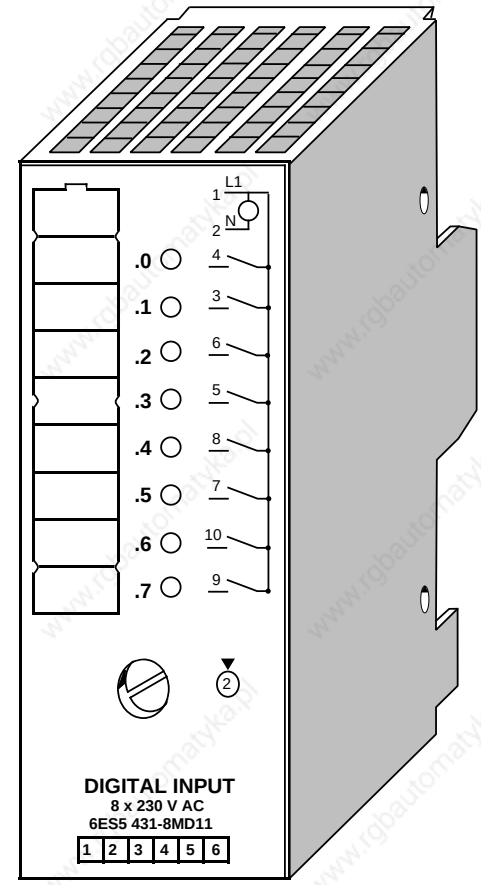


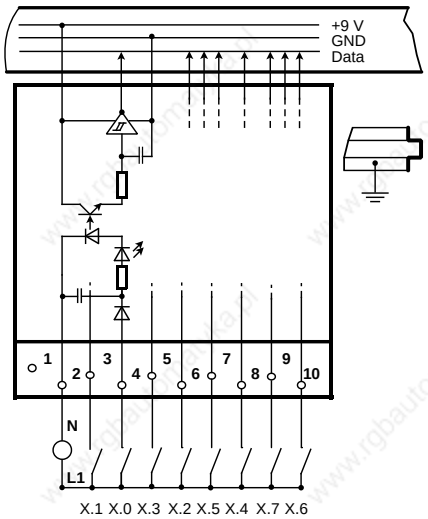




**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Input voltage L1 | | |
| - rated value | | 115 V AC/DC |
| - ”0” signal | | 0 to 40 V |
| - ”1” signal | | 85 to 135 V |
| - frequency | | 47 to 63 Hz |
| Input current at ”1” signal | typ. | 12 mA at 115 V AC |
| | typ. | 2.5 mA at 115 V DC |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 10 ms |
| - from ”1” to ”0” | typ. | 20 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 125 V AC |
| - insulation group | | 2×B |
| - tested with | | 1250 V AC |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 4 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 32 mA |
| Power loss of the module | typ. | 2.5 W |
| Weight | approx. | 260 g (9 oz.) |

## Digital Input Module 8 × 230 V AC (6ES5 431-8MD11)

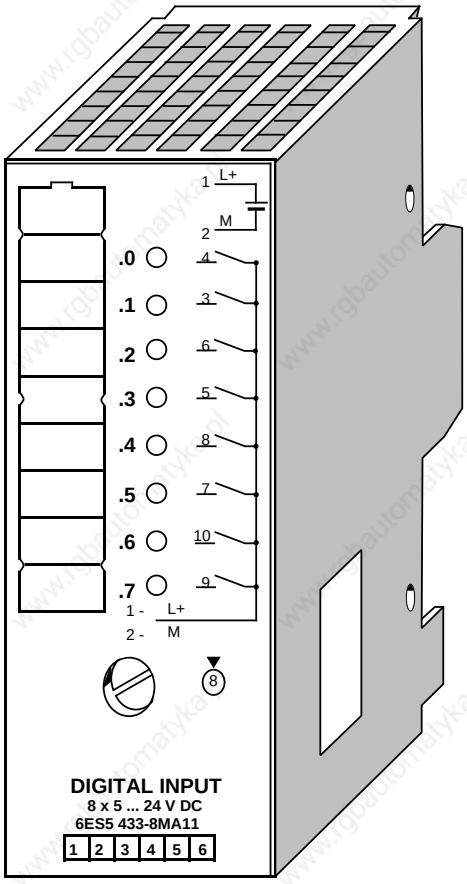


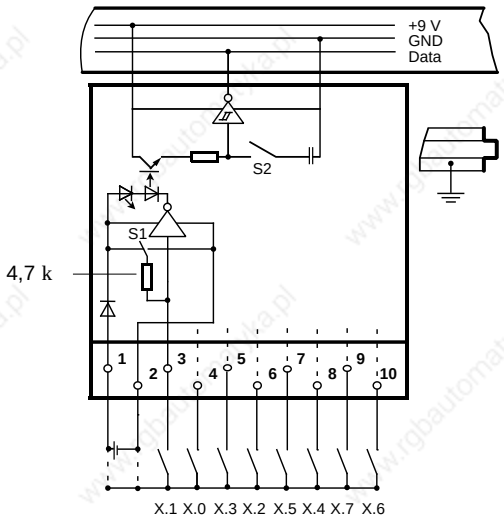




**Technical specifications**

| | | |
|---|---|---|
| Number of inputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Input voltage L1 | | |
| - rated value | | 230 V AC/DC |
| - "0" signal | | 0 to 95 V |
| - "1" signal | | 195 to 253 V |
| - frequency | | 47 to 63 Hz |
| Input current at "1" signal | typ. | 16 mA at 230 V AC |
| | typ. | 1.8 mA at 230 V DC |
| Inherent delay | | |
| - from "0" to "1" | typ. | 10 ms |
| - from "1" to "0" | typ. | 20 ms |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 1500 V AC |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Connection of 2-wire BERO proximity switches | | possible |
| - residual current | | 5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 32 mA |
| Power loss of the module | typ. | 3.6 W |
| Weight | approx. | 260 g (9 oz.) |

## Digital Input Module 8 x 5 to 24 V DC (6ES5 433-8MA11)



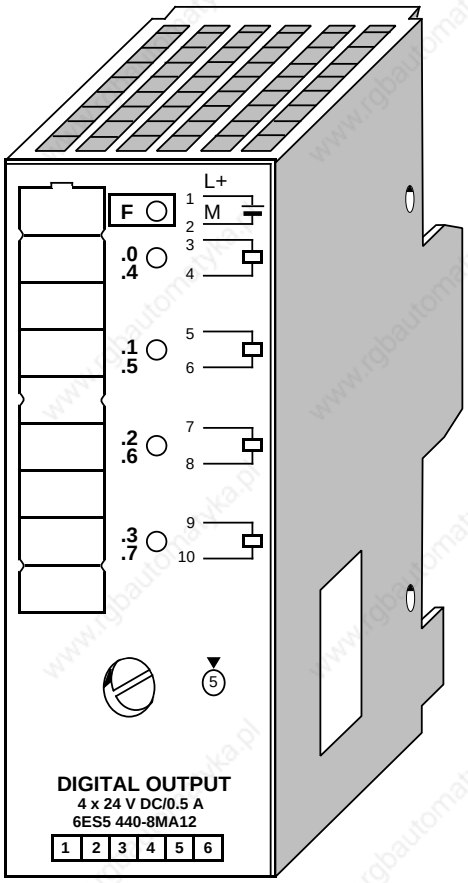

**Technical Specifications**

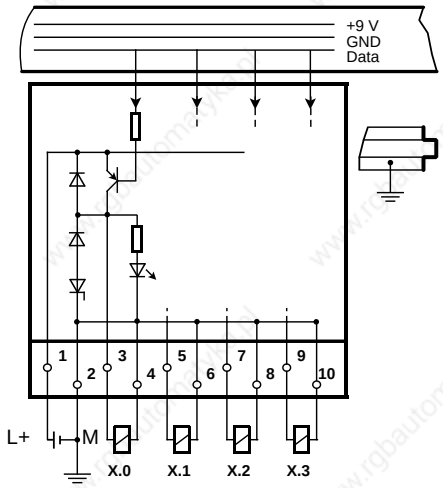
| | | |
|---|---|---|
| Number of inputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Input voltage L+ | | |
| - rated value | | 5 to 24 V DC |
| - "0" signal | | $V_{in}$ approx. 25% L+ |
| - "1" signal | | $V_{in}$ approx. 45% L+ |
| Permissible range | | 4.5 to 30 V |
| Input resistance | | 4.7 k to L+ or M; reversible on the back of the module * |
| The LED displays the evaluated signal | | |
| Inherent delay | approx. | 1 ms or 10 ms; reversible on the back of the module * |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+ 9 V to L+) | | 30 V AC |
| - insulation group | | 2 x B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+ 9 V to ⏚) | | 12 V AC |
| - insulation group | | 2 x B |
| - tested with | | 500 V AC |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Current consumption | | |
| - from + 9 V (CPU) | typ. | 6 mA |
| - from L+ | typ. | 60 mA |
| Power loss of the module | typ. | 2.4 W |
| Weight | approx. | 225 g (8 oz.) |

***** reversible in groups of 8

## 14.6.2 Digital Output Modules

### Digital Output Module 4 × 24 V DC/0.5 A (6ES5 440-8MA12)



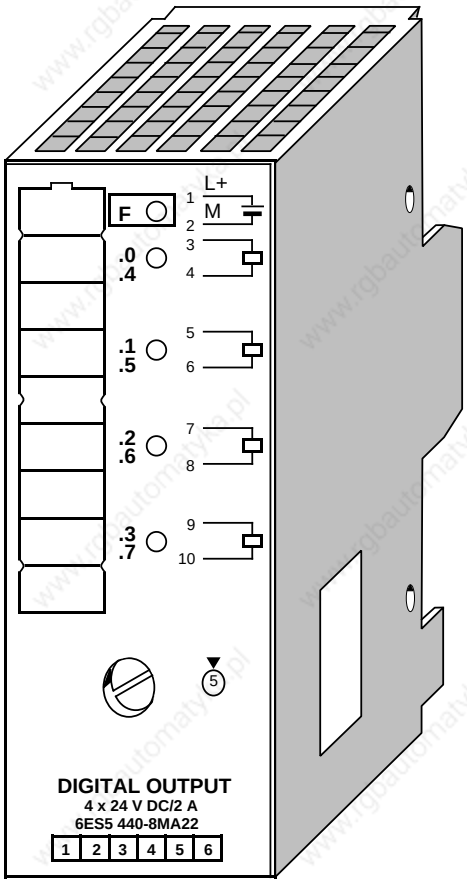



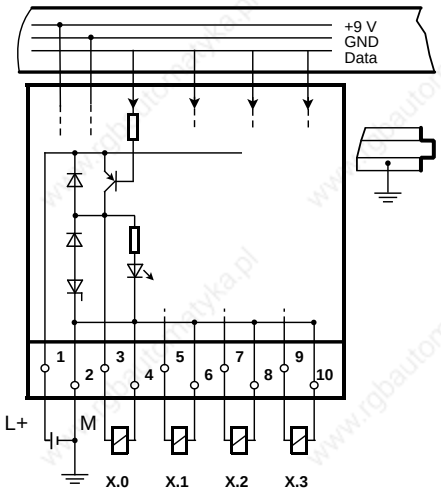

**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 4 |
| Galvanic isolation | | no |
| - in groups of | | 4 |
| Load voltage L+ | | |
| - rated value | | 24 V DC |
| - permissible range (including ripple) | | 20 to 30 V |
| - value at t<0.5 s | | 35 V |
| Output current for "1" signal | | |
| - rated value | | 0.5 A |
| - permissible range | | 5 to 500 mA |
| - lamp load | max. | 5 W |
| Residual current at "0" signal | max. | 0.5 mA |
| Output voltage | | |
| - "1" signal | max. | L+ (- 0.6 V) |
| Short-circuit protection | | short-circuit protected output with autom. switch on when the short-circuit does not exist any more |
| Fault LED (red) | | short-circuit/ no load voltage L+ |
| Error diagnostics | | possible |
| Voltage induced on circuit interruption (internal) limited to | | - 15 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Total permissible current of outputs | | 2 A |
| Driving of digital input | | possible |
| Paralleling of outputs | | possible |
| - maximum current | | 0.8 A |
| Permissible ambient temperature | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage * (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 15 mA |
| - from L+(without load) | typ. | 25 mA |
| Power loss of the module | typ. | 1.5 W |
| Weight | | approx. 200 g (7 oz.) |

* Relevant only for isolated assembly in the ET 100/200U

## Digital Output Module 4 × 24 V DC/2 A (6ES5 440-8MA22)



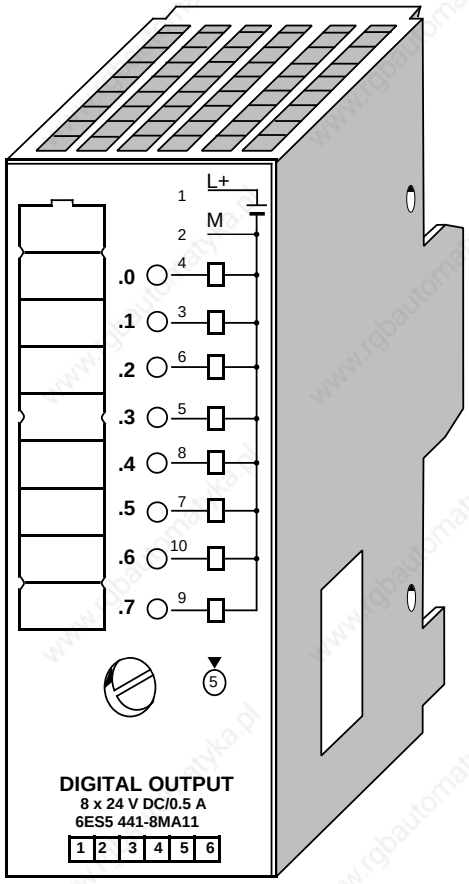



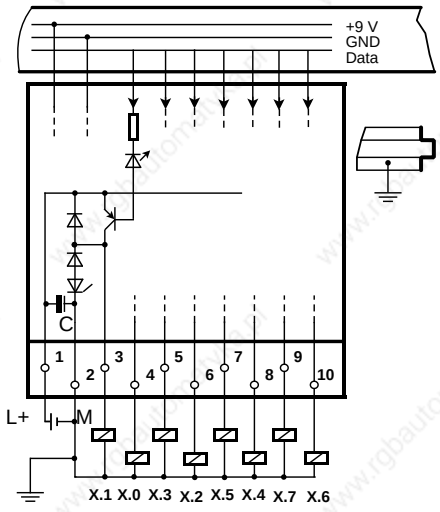

**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 4 |
| Galvanic isolation | | no |
| - in groups of | | 4 |
| Load voltage L+ | | |
| - rated value | | 24 V DC |
| - permissible range | | 20 to 30 V |
| Output current for "1" signal | | |
| - rated value | | 2 A |
| - permissible range | | 5 mA to 2 A |
| - lamp load | max. | 10 W |
| Residual current at "0" signal | max. | 1 mA |
| Output voltage | | |
| - "1" signal | max. | L+ (- 0.8 V) |
| Short-circuit protection | | short-circuit protected output with autom. switch on when the short-circuit does not exist any more |
| Fault LED (red) | | short-circuit/ no load voltage L+ |
| Error diagnostics | | possible |
| Voltage induced on circuit interruption (internal) limited to | | - 15 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Total permissible current of outputs | | 4 A |
| Driving of digital input | | possible |
| Paralleling of outputs | | possible |
| - maximum current | | 3.2 A |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage* (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 15 mA |
| - from L+ (without load) | typ. | 25 mA |
| Power loss of the module | typ. | 3 W |
| Weight | | approx. 200 g (7 oz.) |

* Relevant only for isolated assembly in the ET 100/200U

## Digital Output Module 8 × 24 V DC/0.5 A (6ES5 441-8MA11)




**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 8 |
| Galvanic isolation | | no |
| - in groups of | | 8 |
| Load voltage L+ | | |
| - rated value | | 24 V DC |
| - permissible range (including ripple) | | 20 to 30 V |
| - value at t<0.5 s | | 35 V |

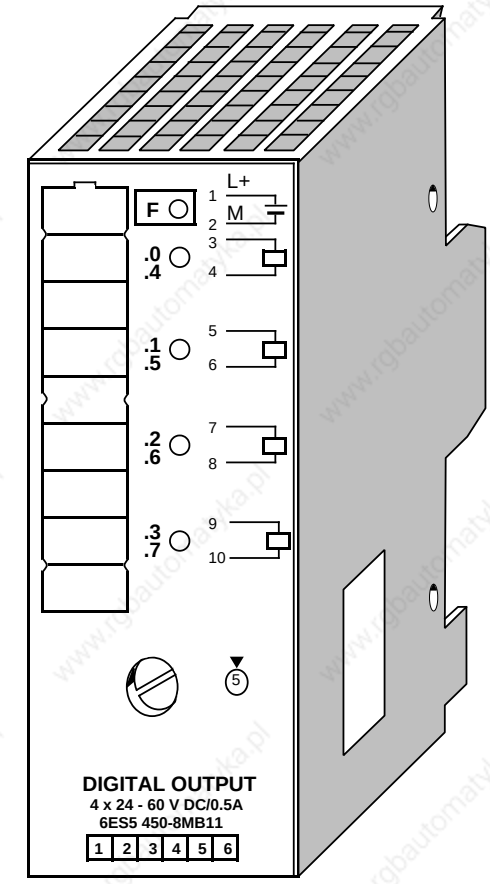
**Warning**

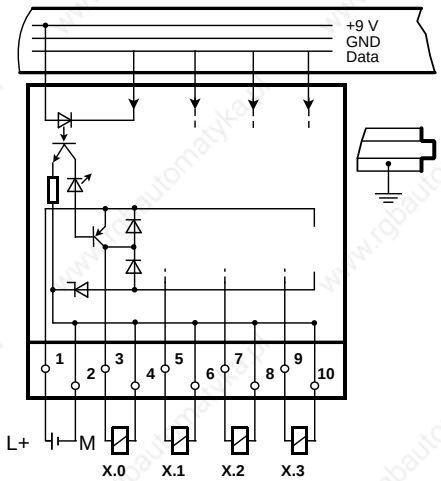
Capacitor remains loaded after switch off of L+

| | | |
|---|---|---|
| Output current for "1" signal | | |
| - rated value | | 0.5 A at 60 °C (140 °F)/ 1 A at 30 °C (86 °F) |
| - permissible range | | 5 mA to 1 A |
| - lamp load | max. | 5 W |
| Residual current at "0" signal | max. | 1.0 mA |
| Output voltage | | |
| - "1" signal | max. | L+ (- 0.5 V) |
| Short-circuit protection | | none |
| Voltage induced on circuit interruption (internal) limited to | | - 15 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Total permissible current of outputs | | 4 A |
| Driving of digital input | | possible |
| Paralleling of 2 outputs | | possible |
| - maximum current | | 0.8 A |
| Permissible ambient temperature of PLC | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage* (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 14 mA |
| - from L+(without load) | typ. | 15 mA |
| Power loss of the module | typ. | 2 W |
| Weight | | approx. 220 g (7.7 oz.) |

* Relevant only for isolated assembly in the ET 100/200U

## Digital Output Module 4 × 24 to 60 V DC/0.5 A (6ES5 450-8MB11)

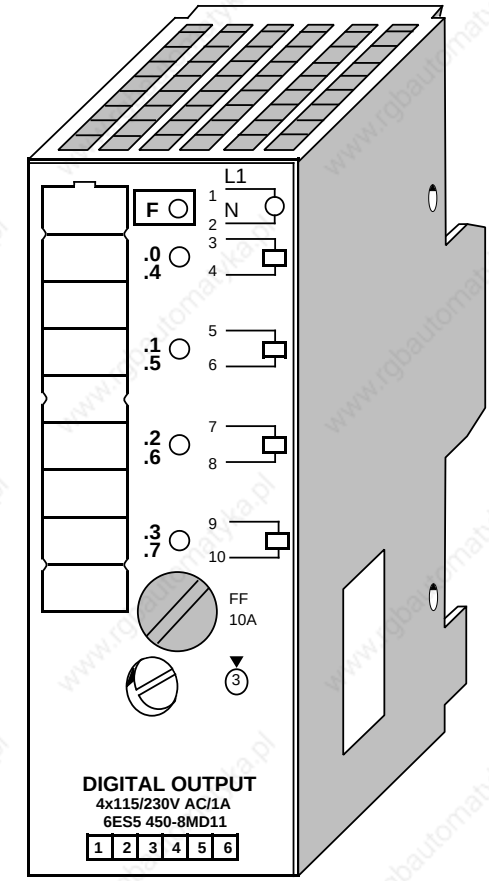


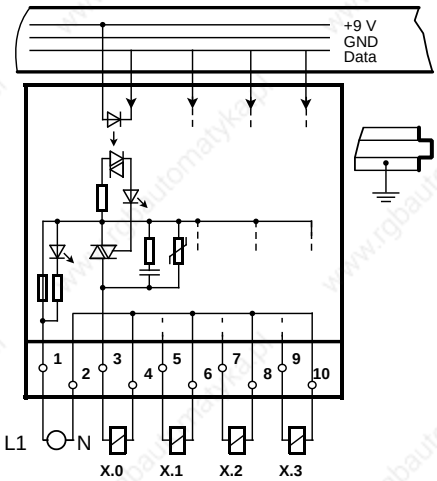




**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 4 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 4 |
| Load voltage L+ | | |
| - rated value | | 24 to 60 V DC |
| - permissible range | | 20 to 72 V |
| Output current for "1" signal | | |
| - rated value | | 0.5 A |
| - permissible range | | 5 mA to 0.5 A |
| - lamp load | max. | 5 to 12 W |
| Residual current at "0" signal | max. | 1 mA |
| Short-circuit protection | | short-circuit protected output with autom. switch on when the short-circuit does not exist any more |
| Fault LED (red) | | short circuit/ no load voltage L+ |
| Error diagnostics | | possible |
| Voltage induced on circuit interruption (internal) limited to | | - 30 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Total permissible current of outputs | | 2 A |
| Driving of digital input | | possible |
| Paralleling of 2 outputs | | possible |
| - maximum current | | 2×0.4 A |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage | | |
| - (+9 V to L+) | | 60 V AC |
| - insulation group | | 2×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 15 mA |
| - from L+ (without load) | typ. | 30 mA (at 60 V) |
| Power loss of the module | typ. | 5 W |
| Weight | | approx. 200 g (7 oz.) |

## Digital Output Module 4 × 115 to 230 V AC/1 A (6ES5 450-8MD11)



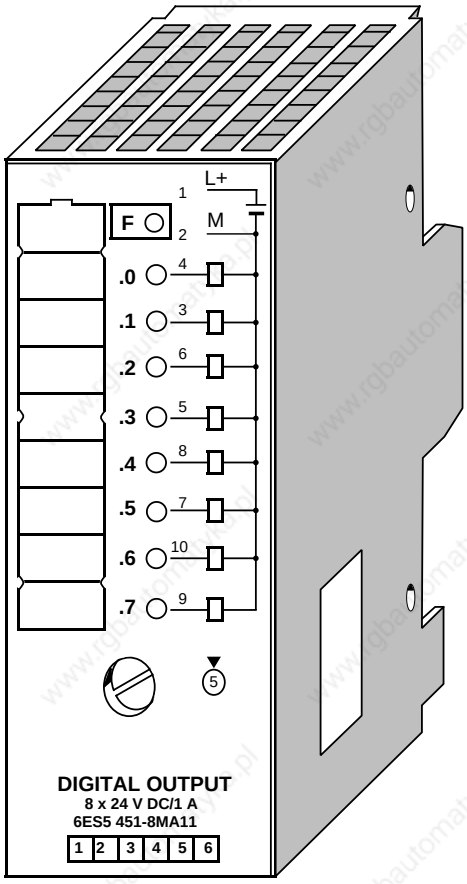



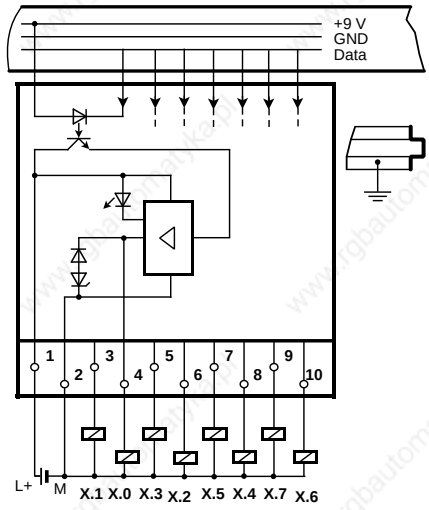

**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 4 |
| Galvanic isolation | | yes |
| - in groups of | | 4 |
| Load voltage L1 | | |
| - rated value | | 115 to 230 V AC |
| - frequency | max. | 47 to 63 Hz |
| - permissible range | | 85 to 264 V |
| Output current for "1"signal | | |
| - rated value | | 1 A |
| - permissible range | | 50 mA to 1 A |
| - lamp load | max. | 25/50 W |
| Contact current closing rating: | | determined by the size of the fuse |
| Residual current at "0" signal | max. | 3/5 mA |
| Output voltage | | |
| - "1" signal | max. | L1 (- 7 V) |
| Signal status display (green LEDs) | | only with load connected |
| Short-circuit protection | | fuse (10 A extra fast) (Wickmann No. 19231, or 6ES5 980-3BC41) |
| Fault LED (red) | | fuse blown * |
| Switching frequency | max. | 10 Hz |
| Permissible current of all outputs | | 4 A |
| Driving of digital input | | possible |
| Paralleling of outputs | | not possible |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 1500 V AC |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 14 mA |
| Power loss of the module | typ. | 3.5 W |
| Weight | approx. | 315 g (11 oz.) |

* Indication only given if load voltage is applied and at least one load is connected

## Digital Output Module 8 x 24 V DC/1 A (6ES5 451-8MA11)

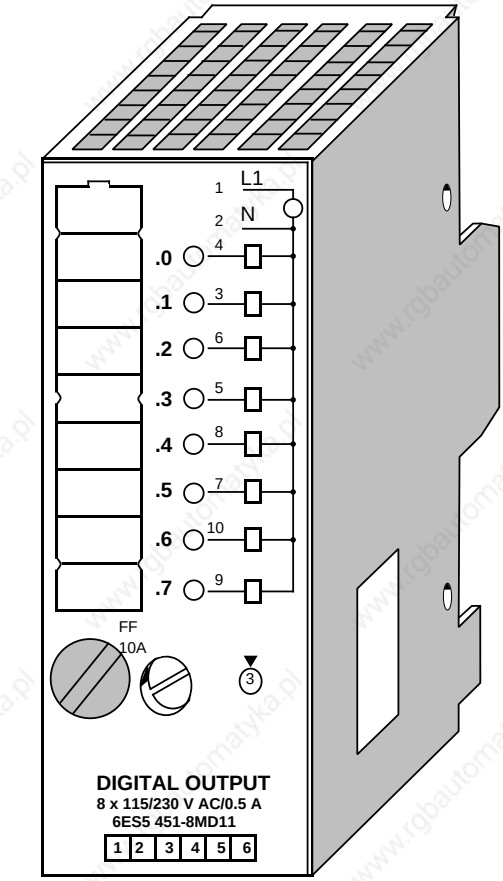


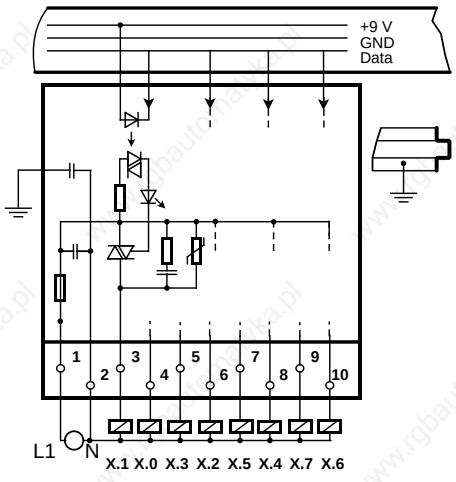




**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Load voltage L+ | | |
| - rated value | | 24 V DC |
| - permissible range (including ripple) | | 20 to 30 V |
| - value at t<0.5 s | | 35 V |
| Output current for "1" signal | | |
| - rated value | | 1 A |
| - permissible range | | 5 mA to 1A |
| - lamp load | max. | 10 W |
| Residual current at "0" signal | max. | 0.5 mA |
| Output voltage | | |
| - at "1" signal | max. | L+ (- 0.6 V) |
| Short-circuit protection | | short-circuit protected output with autom. switch on when the short-circuit does not exist any more |
| Fault LED (red) | | short-circuit |
| Voltage induced on circuit interruption (internal) limited to | | -15 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Permissible current of all outputs | | 6 A |
| Driving of digital input | | possible |
| Paralleling of 2 outputs | | paarweise possible |
| - maximum current | | 1.8 A |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+ 9 V to L+) | | 24 V AC |
| - insulation group | | 2 x B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+ 9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 35 mA |
| - from L+ (without load) | typ. | 50 mA |
| Power loss of the module | typ. | 3 W |
| Weight | approx. | 230g (8 oz.) |

## Digital Output Module 8 × 115 to 230 V AC/0.5 A (6ES5 451-8MD11)

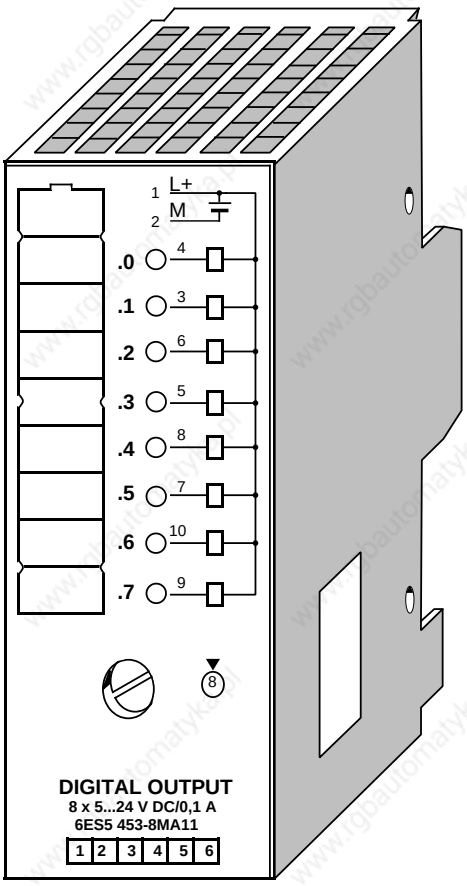


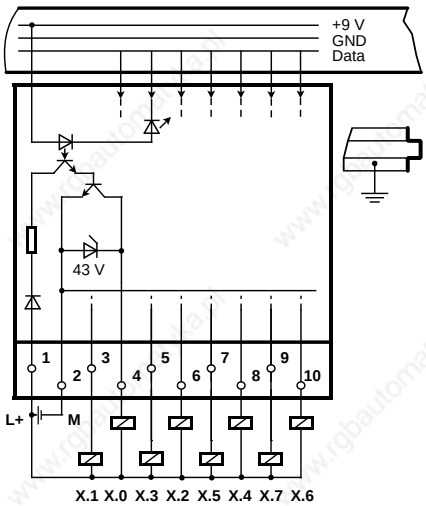




**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 8 |
| Galvanic isolation | | yes (optocoupler) |
| - in groups of | | 8 |
| Load voltage L1 | | |
| - rated value | | 115 to 230 V AC |
| - frequency | max. | 47 to 63 Hz |
| - permissible range | | 85 to 264 V |
| Output current for "1" signal | | |
| - rated value | | 0.5 A |
| - permissible range | | 50 mA to 0.5 A |
| - lamp load | max. | 25/50 W |
| Contact current closing rating: | | determined by the size of the fuse |
| Residual current at "0" signal | max. | 3/5 mA |
| Output voltage | | |
| - at "1" signal | max. | L1 (-7 V) |
| Signal Status Display (green LEDs) | | only with load connected |
| Short-circuit protection | | fuse (10 A extra fast) (Wickmann No. 19231, or 6ES5 980-3BC41) |
| Switching frequency | max. | 10 Hz |
| Permissible current of all outputs | | 4 A |
| Driving of digital input | | possible |
| Paralleling of outputs | | not possible |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to L1) | | 250 V AC |
| - insulation group | | 2×B |
| - tested with | | 1500 V AC |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 25 mA |
| Power loss of the module | typ. | 3.5 W |
| Weight | approx. | 270 g (9 oz.) |

## Digital Output Module 8 × 5 to 24 V DC/0.1 A (6ES5 453-8MA11)



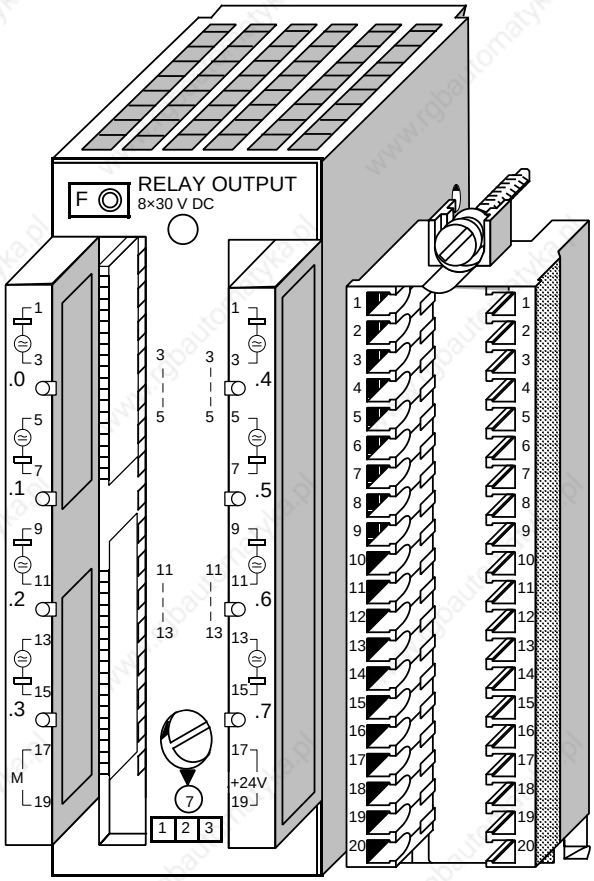



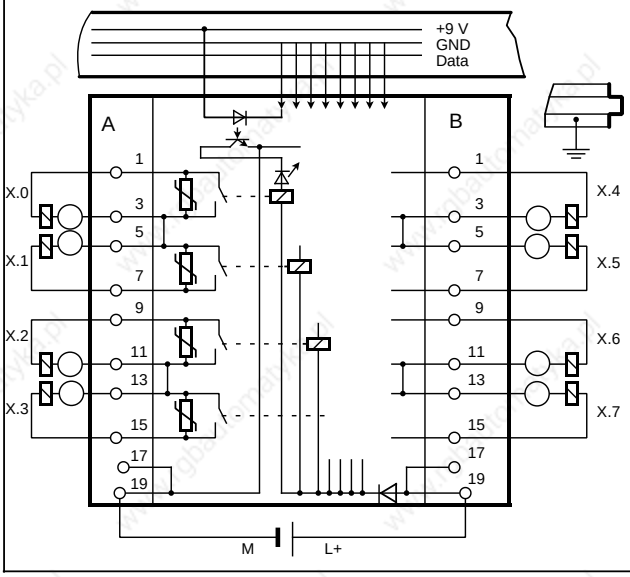

**Technical specifications**

| | | |
|---|---|---|
| Number of outputs | | 8 |
| Galvanic isolation | | yes |
| - in groups of | | 8 |
| Load voltage L+ | | |
| - rated value | | 5 to 24 V DC |
| - permissible range (including ripple) | | 4.75 to 30 V |
| - value at t<0.5 s | | 35 V |
| Output voltage | | TTL-compatible [1] |
| Output current for "1" signal | | |
| - rated value | | 100 mA |
| Short-circuit protection | | none |
| Voltage induced on circuit interruption (internal) limited to | | - 19 V (at 24 V) |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |
| Paralleling of 2 outputs | | possible |
| Permissible ambient temperature of module | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 20 mA |
| - from L+ (without load) | typ. | 28 mA |
| Power loss of the module | typ. | 1 W |
| Weight | approx. | 220 g (8 oz.) |

[1] transistor with open collector, switching to M potential

**Relay Output Module 8 x 30 V DC/230 V AC** **(6ES5 451-8MR12)**
**Crimp Snap-in Connector, 40-pin** **(6ES5 490-8MA13/-8MA03)**
**Screw Plug Connector, 20-pin** **(6ES5 490-8MB21)**
**Screw Plug Connector, 40-pin** **(6ES5 490-8MB11)**

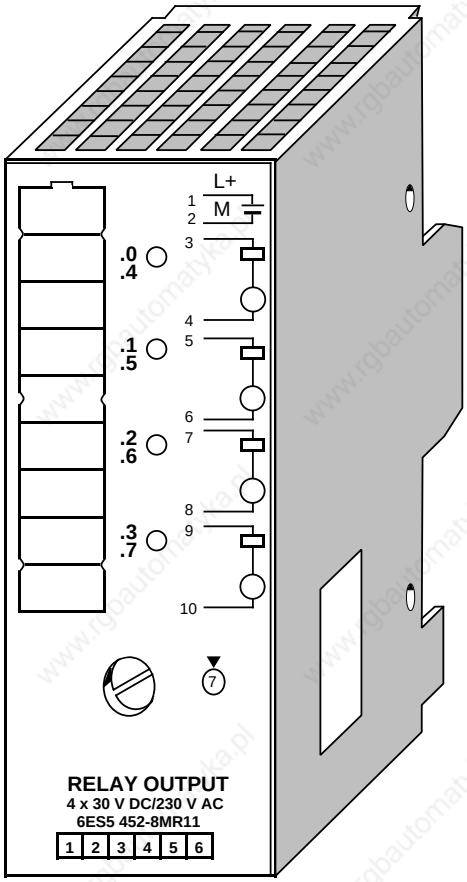


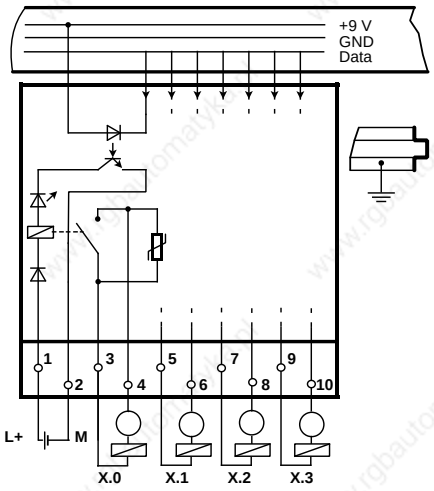




**Technical specifications**

| | | |
|---|---|---|
| Outputs | | 8 relay outputs, contact switching varistor SIOV-S07-K275 |
| Galvanic isolation<br>- in groups of | | yes<br>2 with signal status display |
| Continuous current $I_{th}$ | | 3 A |
| Relay type | | Dold OW 5699 |
| Switching capacity of the contacts<br>- resistive load | max. | 3 A at 250 V AC<br>1.5 A at 30 V DC |
| - inductive load | max. | 0.5 A at 250 V AC<br>0.5 A at 30 V DC |
| Operating cycles of the contacts according to VDE 0660, part 200<br>- AC - 11<br>- DC - 11 | | <br>$1 \times 10^6$<br>$0.5 \times 10^6$ |
| Switching frequency<br>Fault LED (red) | max. | 10 Hz<br>no input voltage |
| Permissible ambient temperature of module<br>- horizontal arrangement<br>- vertical arrangement | | <br>0 to 60 °C (32 to 140 °F)<br>0 to 40 °C (32 to 104 °F) |
| Length of cable<br>- unshielded<br>Insulation rating | max. | <br>100 m (330 ft.)<br>VDE 0160 |
| Rated insulation voltage (+ 9 V to L 1)<br>- insulation group<br>- tested with | | 250 V AC<br>2 x B<br>1500 V AC |
| Rated insulation voltage (+ 9 V to ⏚)<br>- insulation group<br>- tested with | | 12 V AC<br>1 x B<br>500 V AC |
| Rated insulation voltage (between contacts)<br>- insulation group<br>- tested with | | 250 V AC<br>2 x B<br>1500 V AC |
| Supply voltage L+ (for the relay)<br>- rated value<br>- ripple $V_{PP}$<br>- permissible range (ripple included)<br>- value to t <0.5 s | <br>max.<br><br> | 24 V DC<br>3.6 V<br>20 to 30 V<br>35 V |
| Current consumption<br>- from + 9 V (CPU)<br>- from L+ | <br>typ.<br>typ. | <br>30 mA<br>70 mA |
| Power loss of the module | typ. | 1.6 W |
| Weight | approx. | 300 g (11 oz.) |

## Relay Output Module 4 x 30 V DC/230 V AC (6ES5 452-8MR11)



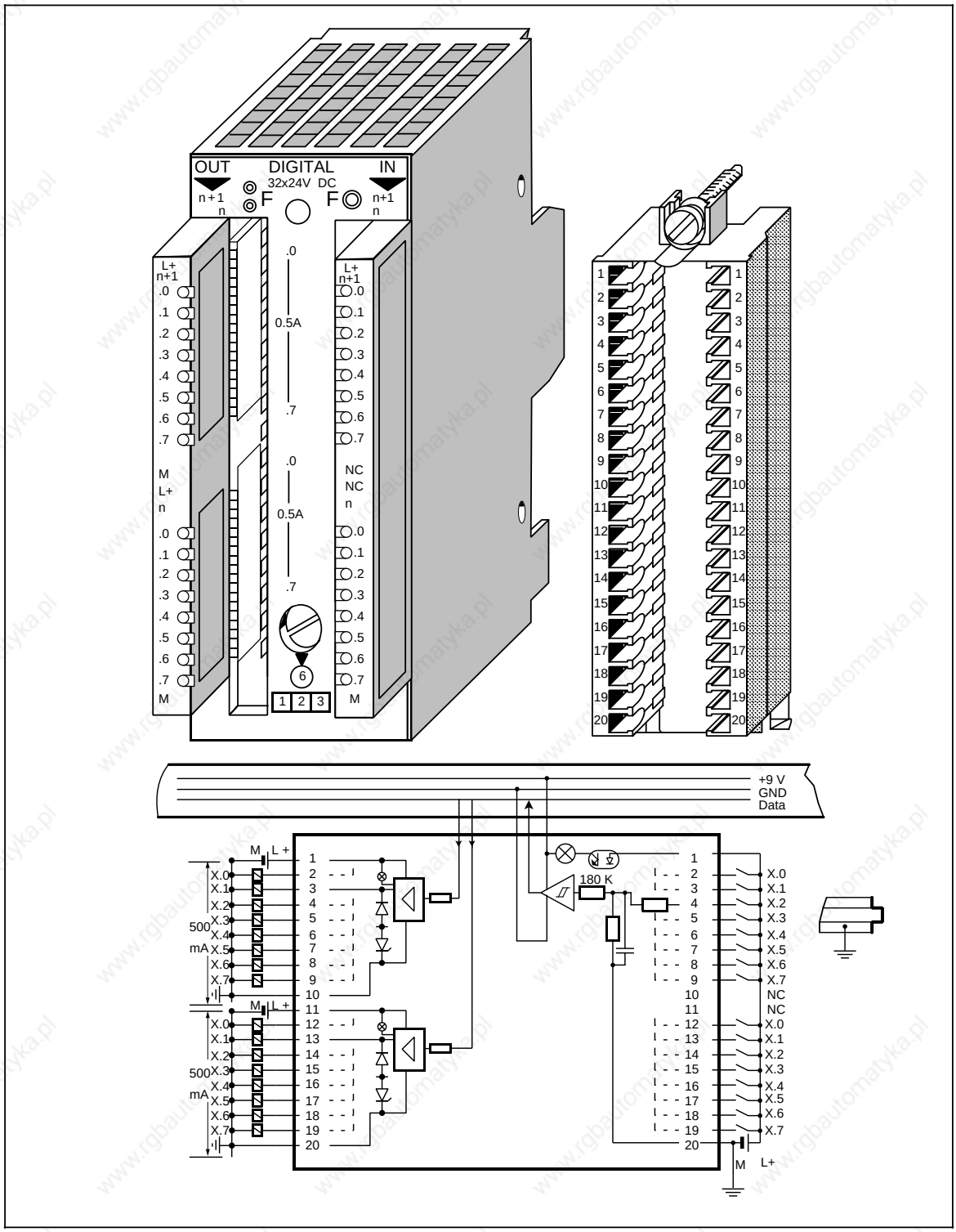




**Technical specifications**

| | | |
|---|---|---|
| Outputs | | 4 relay outputs, contact switching varistor SIOV-S07-K275 |
| Galvanic isolation<br>- in groups of | | yes (optocoupler)<br>1 |
| Continuous current $I_{th}$ | | 5 A |
| Relay type | | Siemens V 23127-D 0006-A402 |
| Switching capacity of the contacts<br>- resistive load<br>- inductive load | <br>max.<br>max. | <br>5 A at 250 V AC<br>2.5 A at 30 V DC<br>1.5 A at 250 V AC<br>0.5 A at 30 V DC |
| Operating cycle of the contacts according to VDE 0660, part 200<br>- AC-11<br>- DC-11 | | <br>$1.5 \times 10^6$<br>$0.5 \times 10^6$ |
| Switching frequency | max. | 10 Hz |
| Permissible ambient temperature of module<br>- horizontal arrangement<br>- vertical arrangement | | <br>0 to 60 °C (32 to 140 °F)<br>0 to 40 °C (32 to 104 °F) |
| Length of cable<br>- unshielded | <br>max. | <br>100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+ 9 V to L1)<br>- insulation group<br>- tested with | | 250 V AC<br>2 x B<br>1500 V AC |
| Rated insulation voltage (+ 9 V to ⏚)<br>- insulation group<br>- tested with | | 12 V AC<br>1 x B<br>500 V AC |
| Rated insulation voltage (between contacts)<br>- insulation group<br>- tested with | | 250 V AC<br>2 x B<br>1500 V AC |
| Supply voltage L+ (for the relay)<br>- rated value<br>- ripple $V_{pp}$<br>- permissible range (ripple included)<br>- value at t<0.5 s | <br>max. | <br>24 V DC<br>3.6 V<br>20 to 30 V<br>35 V |
| Current consumption<br>- from + 9 V (CPU)<br>- from L+ | <br>typ.<br>typ. | <br>14 mA<br>100 mA |
| Power loss of the module | typ. | 2 W |
| Weight | approx. | 240 g (8 oz.) |

## 14.6.3 Digital Input/Output Modules

**Digital Input/Output Module with LED Display** **(6ES5 482-8MA13)**
**Crimp Snap-in Connector, 40-pin** **(6ES5 490-8MA13/-8MA03)**
**Screw Plug Connector, 40-pin** **(6ES5 490-8MB11)**



## Digital Input/Output Module with LED Display (continued) (6ES5 482-8MA13)

**Technical specifications**

| | | |
|---|---|---|
| Cable length | | |
| - unshielded | | 100 m (330 ft.) |
| Rated insulation voltage | | |
| (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1 x B |
| Power loss of the module | typ. | 4.5 W |
| Weight | approx. | 190 g (7 oz.) |

**Input side**

| | | |
|---|---|---|
| Number of inputs | | 16 |
| Galvanic isolation | | no |
| - in groups of | | 16 |
| Input voltage L+ | | |
| - rated value | | 24 V DC |
| - for ”0” signal | | 0 to 5 V |
| - for”1” signal | | 13 to 30 V |
| Input current for ”1” signal | typ. | 4.5 mA |
| Inherent delay | | |
| - from ”0” to ”1” | typ. | 4 ms |
| - from ”1” to ”0” | typ. | 3 ms |
| Fault LED (red) | | indicates interruption of L+/M supply |
| Connection of two-wire BERO proximity switches | | possible |
| - residual current | | 1.5 mA |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 50 mA |

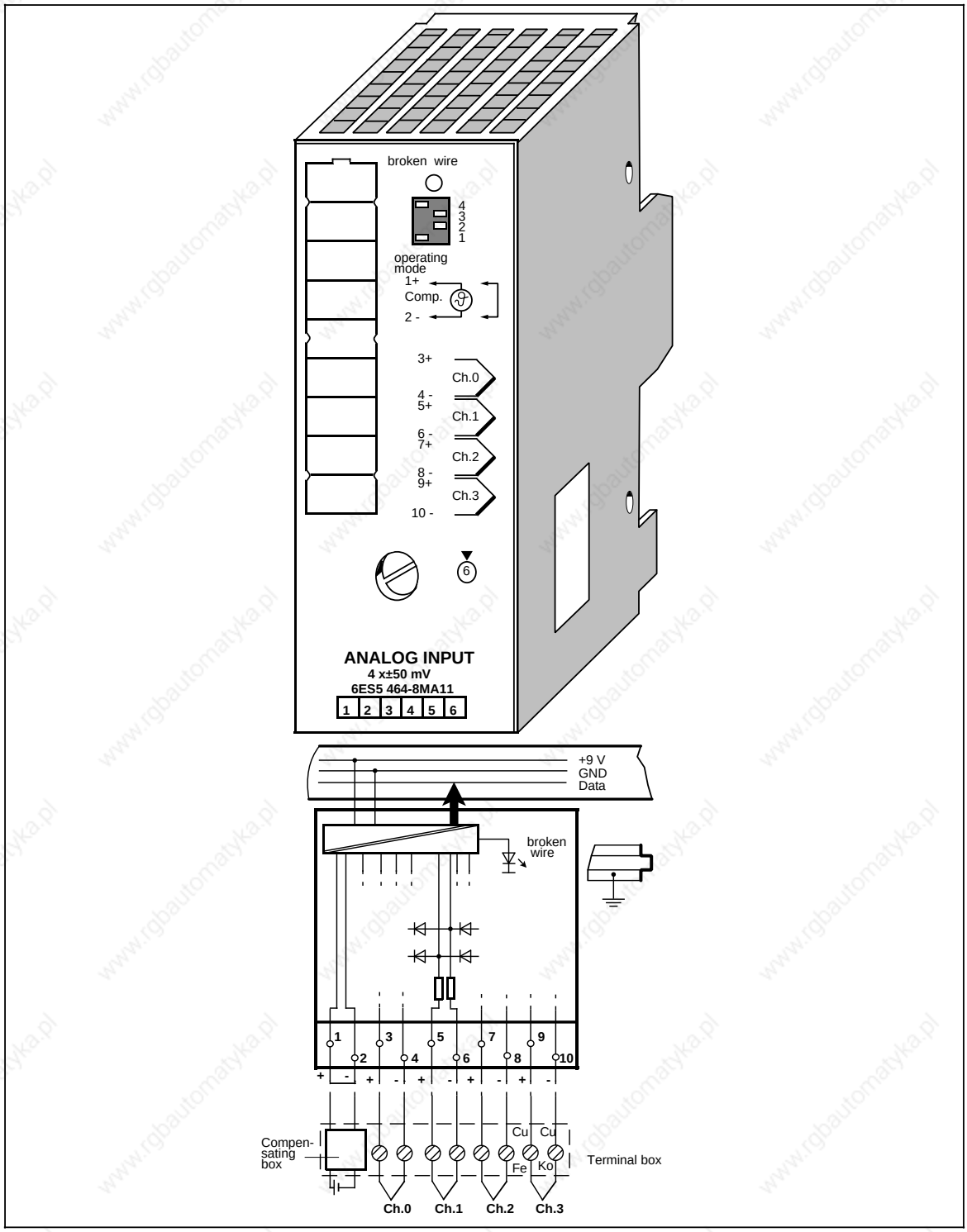
**Output side**

| | | |
|---|---|---|
| Number of outputs | | 16 |
| Galvanic isolation | | no |
| - in groups of | | 8 |
| Load voltage L+ | | |
| - rated value | | 24 V DC |
| - permissible range (ripple included) | | 20 to 30 V |
| - value at t<0.5 s | | 35 V |
| Output current $I_N$ for ”1” signal | | |
| - rated value | | 500 mA |
| - permissible range | | 5 to 500 mA |
| Residual current for ”0” signal | max. | 0.5 mA |
| Short-circuit protection | | yes |
| Short-circuit indication | | red LED |
| Output voltage for ”1” signal | | L+(- 0.6 V) |
| Voltage induced on circuit interruption (internal) limited to | | - 15 V |
| Switching frequency with | | |
| - resistive load | | 100 Hz |
| - inductive load | | 2 Hz |
| Permissible total current of the outputs | | 6 A |
| Driving of a digital input | | possible |
| Paralleling of outputs | | possible in pairs |
| - maximum current | | $(0.8 \times I_N)$ |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 10 mA |
| - from L+ (without load) | typ. | 100 mA |
| Lamp load | max. | 5 W |

## 14.7 Analog Modules

### 14.7.1 Analog Input Modules

**Analog Input Module 4 x ±50 mV** **(6ES5 464-8MA11)**



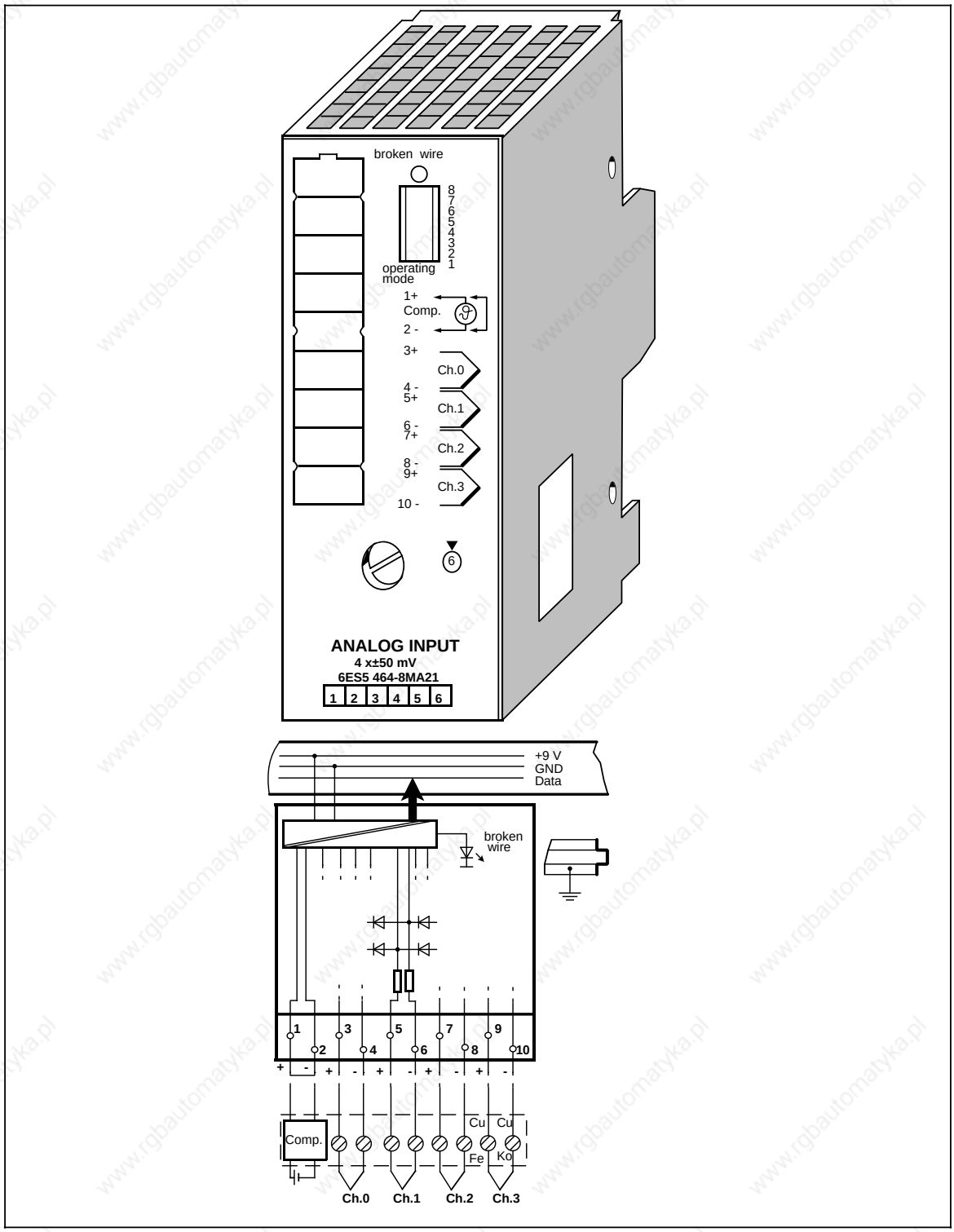
## Analog Input Module 4 × ±50 mV (continued) (6ES5 464-8MA11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | ±50 V |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 10 M |
| Connection method of sensors | | two-wire connection |
| Digital representation of input signal | | 12 bits+sign (2048 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz<br>max. 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz<br>max. 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ±1V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 24 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | yes (selectable) |
| - general indication of wire break | | red LED |

| | | |
|---|---|---|
| Noise suppression for f=nx (50/60 Hz±1%); n=1, 2, ... | | |
| - common-mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise <rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.15 % |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.4 % |
| Single errors | | |
| - linearity | | ±0.05 % |
| - tolerance | | ±0.05 % |
| - polarity reversal error | | ±0.05 % |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Length of cable | | |
| - shielded | max. | 50 m (164 ft.) |
| Supply voltage L+ | | none |
| Connection of compensating box | | possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to+9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from+9 V (CPU) | typ. | 70 mA |
| Power loss of the module | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 x ±50 mV (6ES5 464-8MA21)**



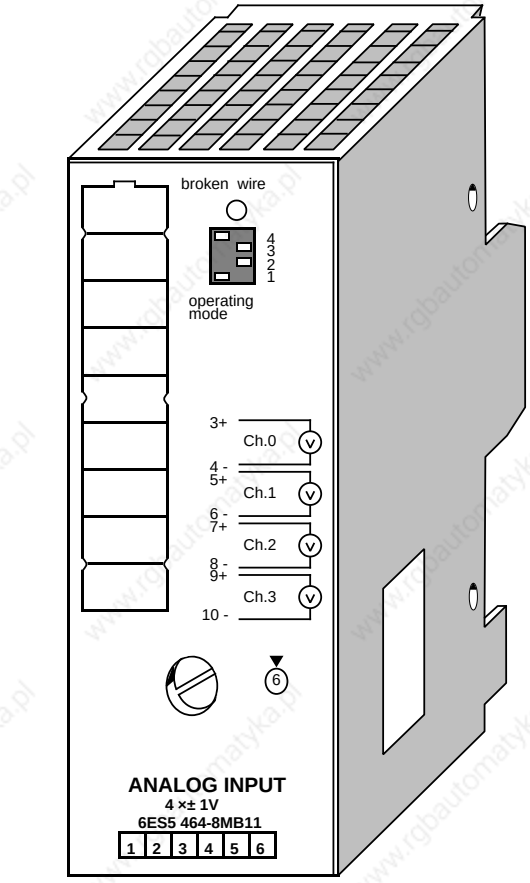
## Analog Input Module 4 x ±50 mV (continued) (6ES5 464-8MA21)

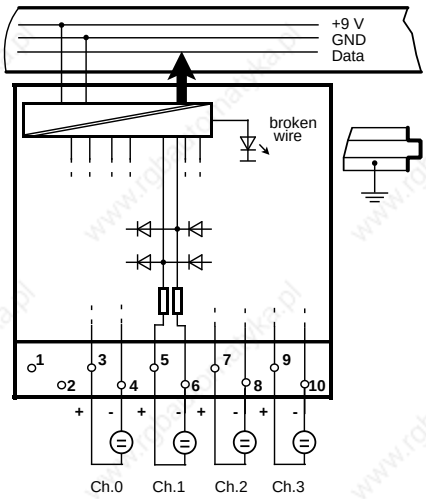
**Technical specifications**

| | | |
|---|---|---|
| Input range (rated values) | | ± 50 mV |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 10 M |
| Connection method of sensors | | two-wire connection |
| Digital representation of input signal | | 12 bits + sign (2048 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ±1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 24 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | yes (selectable) |
| - general indication of wire break | | red LED |

| | | |
|---|---|---|
| Noise suppression for f = nx (50/60 Hz±1%) n = 1, 2, ... | | |
| - common mode rejection ($V_{pp}$ = 1 V) | | min. 86 dB |
| - series mode rejection (peak value of noise < rated value of input range) | | min. 40 dB |
| Basic error limits | | ±0.15% |
| Operating error limits (0 to 60 °C) (32 to 140 °F) | | ±0.4% |
| Single errors | | |
| - linearity | | ±0.05% |
| - tolerance | | ±0.05% |
| - polarity reversal error | | ±0.05% |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Linearization exactness for rated range (for types J,K,L) | | ±1 °C (1.8 °F) |
| Characteristic linearization for the following thermoelements | | |
| - Nickel-Chromium/ Nickel-Aluminium (Type K) | | IEC 584 |
| - Iron/Copper-Nickel (Type J) | | IEC 584 |
| - Iron/Copper-Nickel (Type L) | | DIN 43710 |
| Length of cable | | |
| - shielded | max. | 50 m (164 ft.) |
| Supply voltage L+ | | none |
| Connection of compensating box | | possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V DC |
| Rated insulation voltage (inputs to + 9 V) | | 60 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from + 9 V (CPU) | typ. | 100 mA |
| Power loss of the module | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 x ±1 V** **(6ES5 464-8MB11)**



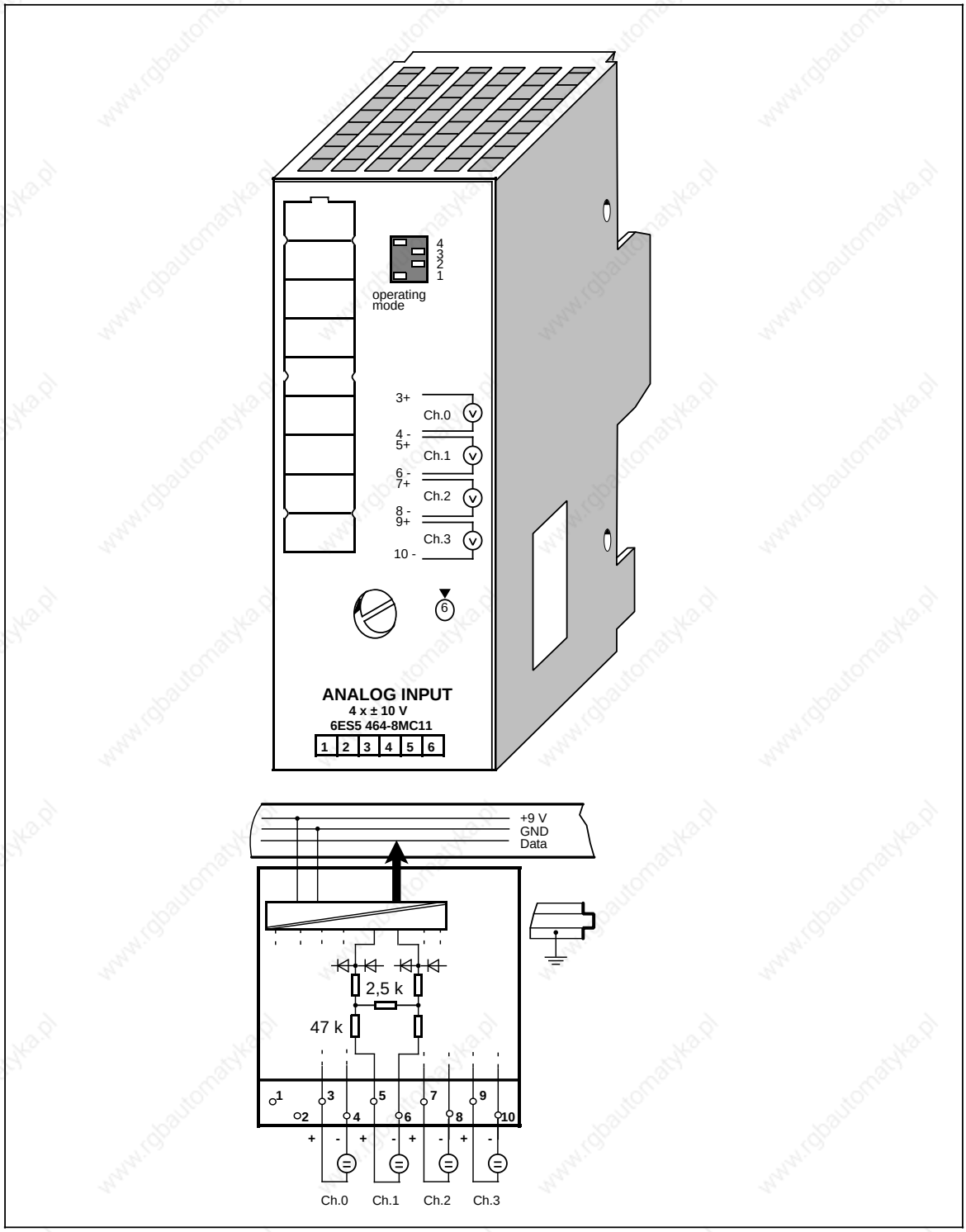
## Analog Input Module 4 x ±1 V (continued) (6ES5 464-8MB11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | ± 1 V |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 10 M |
| Connection method of sensors | | two-wire connection |
| Digital representation of input signal | | 12 bits+sign (2048 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ± 1V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 24 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | yes (selectable) |
| - general indication of wire break | | red LED |

| | | |
|---|---|---|
| Noise suppression for f=nx (50/60 Hz±1%); n=1, 2, ... | | |
| - common-mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise < rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.1 % |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.35 % |
| Single errors | | |
| - linearity | | ±0.05 % |
| - tolerance | | ±0.05 % |
| - polarity reversal error | | ±0.05 % |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ | | none |
| Connection of compensating box | | not possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to +9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from+9 V (CPU) | typ. | 70 mA |
| Power loss of the module | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 x ±10 V** **(6ES5 464-8MC11)**



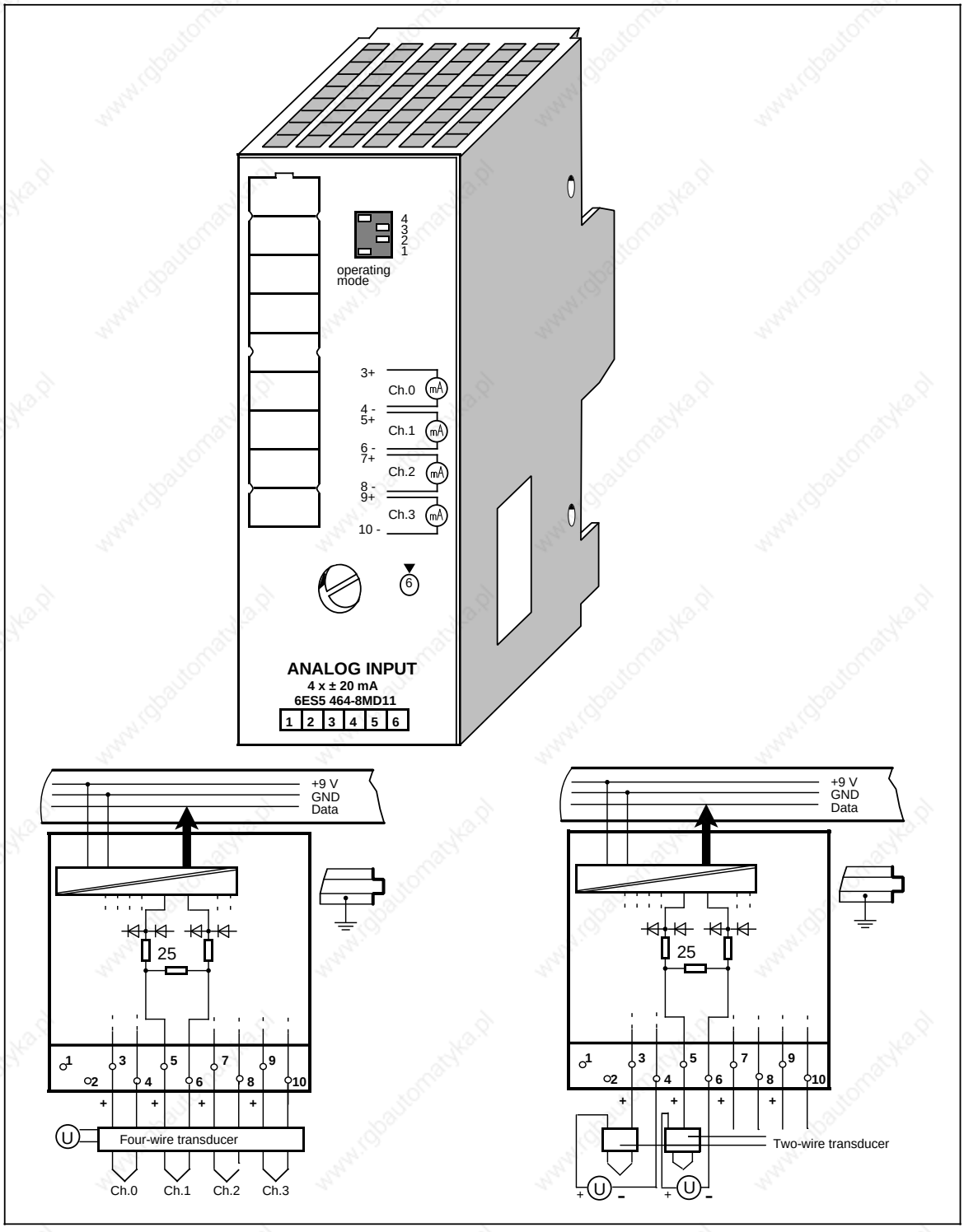
## Analog Input Module 4 x ±10 V (continued) (6ES5 464-8MC11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | ±10 V |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 50 k |
| Connection method of sensors | | two-wire connection |
| Digital representation of input signal | | 12 bits+sign (2048 units =rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ±1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 50 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | no |
| - general indication of wire break | | no |

| | | |
|---|---|---|
| Noise suppression for f=nx (50/60 Hz±1%); n=1,2, ... | | |
| - common-mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise < rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.2 % |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.45 % |
| Single errors | | |
| - linearity | | ±0.05 % |
| - tolerance | | ±0.05 % |
| - polarity reversal error | | ±0.05 % |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ | | none |
| Connection of compensating box | | not possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to +9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 70 mA |
| Power loss of the module | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 x ±20 mA** **(6ES5 464-8MD11)**



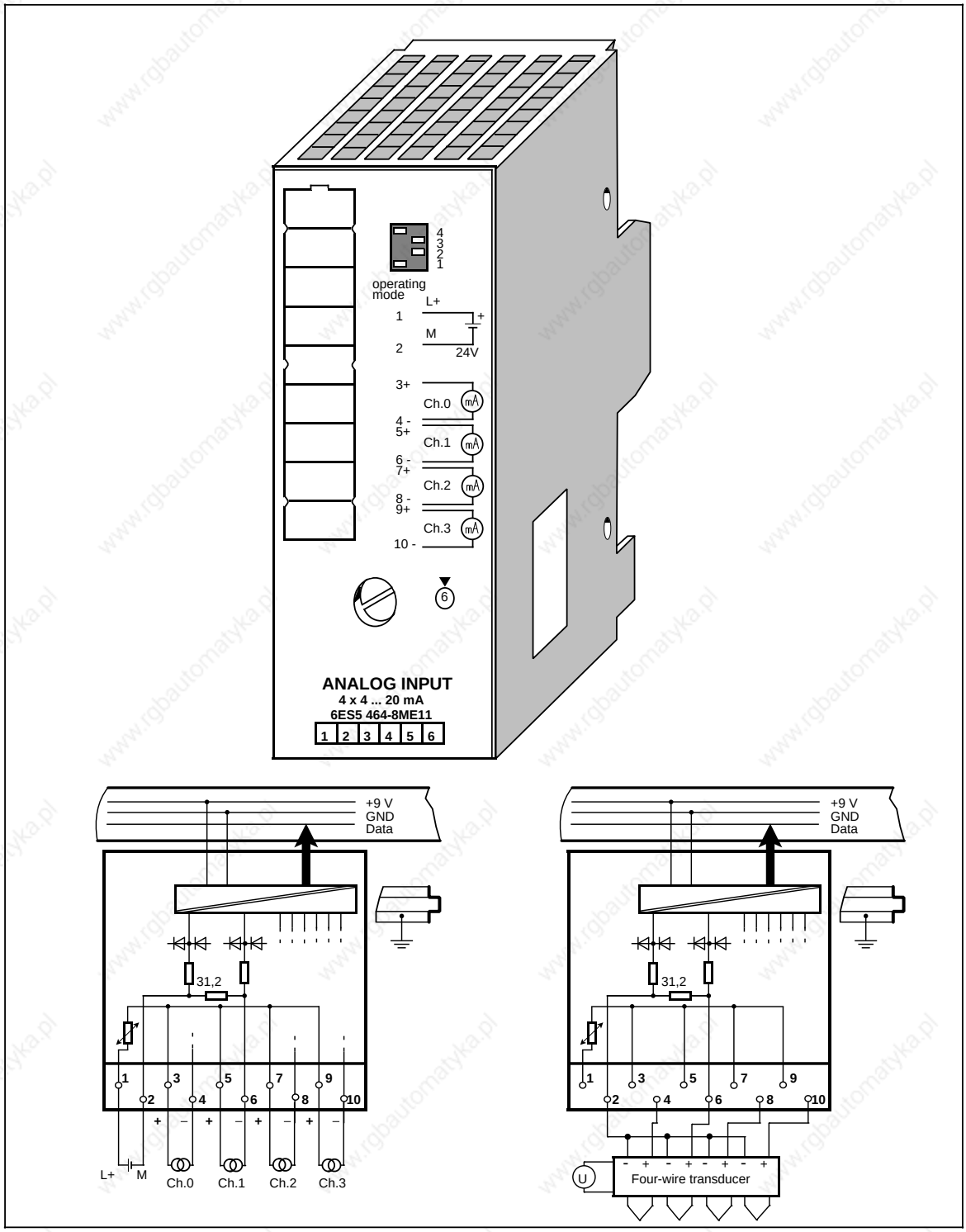
## Analog Input Module 4 x ±20 mA (continued) (6ES5 464-8MD11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | ±20 mA |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 25 |
| Connection method of sensors | | two-wire connection |
| Digital representation of input signal | | 12 bits+sign (2048 units =rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ±1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 80 mA |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | no |
| - general indication of wire break | | no |

| | | |
|---|---|---|
| Noise suppression for f=nx (50/60 Hz±1%); n=1,2, ... | | |
| - common-mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise <rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.2 % |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.45 % |
| Single errors | | |
| - linearity | | ±0.05 % |
| - tolerance | | ±0.05 % |
| - polarity reversal error | | ±0.05 % |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ | | none |
| Connection of compensating box | | not possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to +9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 70 mA |
| Power loss of the module | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 x 4 to 20 mA** **(6ES5 464-8ME11)**



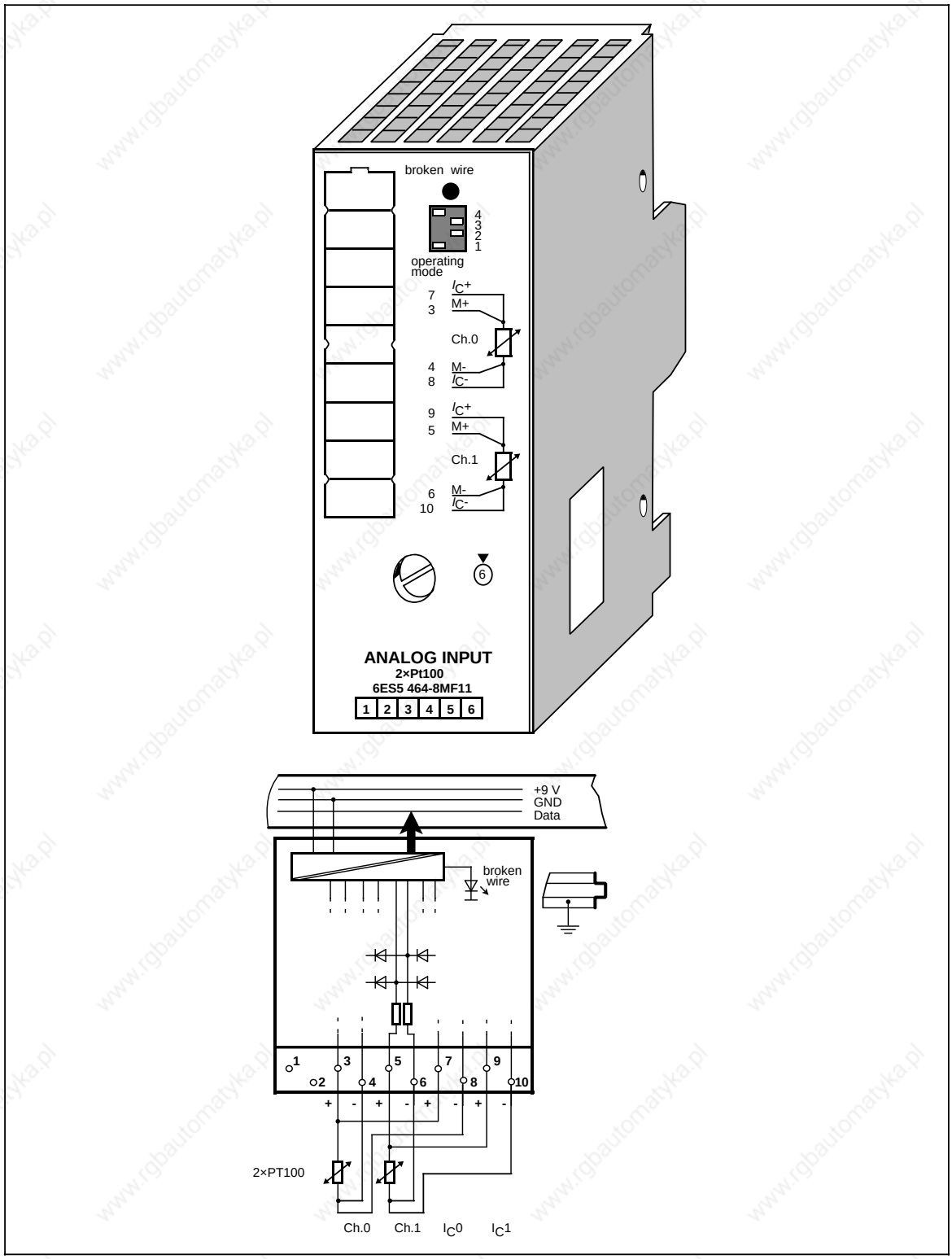
## Analog Input Module 4 x 4 to 20 mA (continued) (6ES5 464-8ME11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | 4 to 20 mA |
| Number of inputs | | 1, 2 or 4 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 31.25 |
| Connection method of sensors | | two-wire connection for 2/4 wire transducers |
| Digital representation of input signal | | 12 bits+sign (2048 units =rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ±1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 80 mA |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | no |
| - general indication of wire break | | no |

| | | |
|---|---|---|
| Noise suppression for f=nx (50/60 Hz±1%); n=1, 2, ... | | |
| - common-mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise <rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.15 % |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.4 % |
| Single errors | | |
| - linearity | | ±0.05 % |
| - tolerance | | ±0.05 % |
| Temperature error | | |
| - final value | | ±0.01 %/K |
| - zero point | | ±0.002 %/K |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ for 2-wire transducers | | |
| - rated value | | 24 V DC |
| - ripple $V_{pp}$ | | 3.6 V |
| - permissible range | | 20 to 30 V |
| Connection of compensating box | | not possible |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to +9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 70 mA |
| - from L+ | typ. | 80 mA |
| Power loss of the module | | |
| - for 2-wire transducers | typ. | 1.0 W |
| - for 4-wire transducers | typ. | 0.7 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 2 x PT 100/±500 mV** **(6ES5 464-8MF11)**



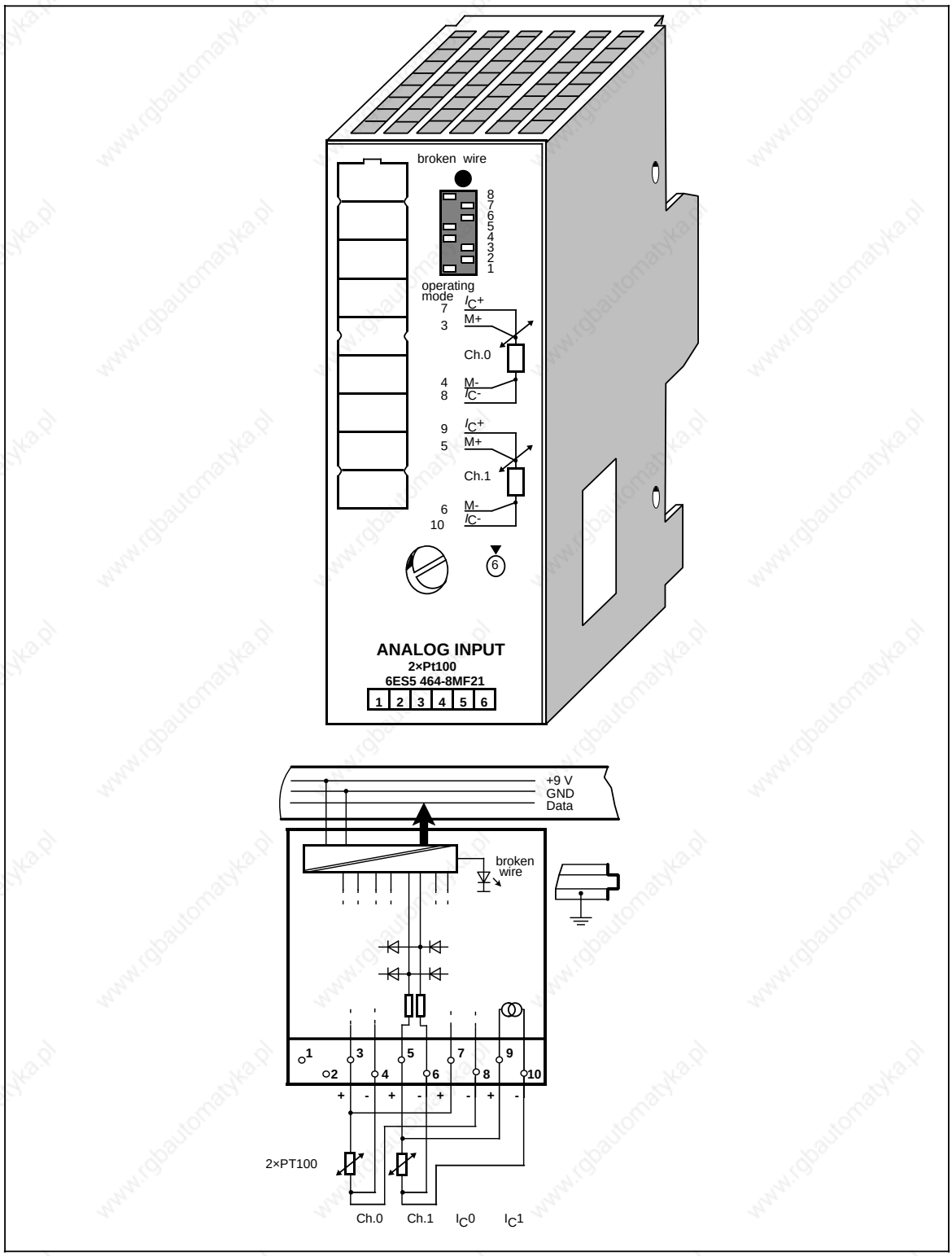
## Analog Input Module 2 × PT 100/±500 mV (continued) (6ES5 464-8MF11)

**Technical specifications**

| | | |
|---|---|---|
| Input range (rated values) | | |
| - resistance sensor (PT 100) | | 0 to 200 (max. 400 ) |
| - voltage sources | | ± 500 mV |
| Number of inputs | | 1 or 2 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 10 M |
| Connection method of sensors | | two or four-wire connection |
| Digital representation of input signal | | 12 bits + sign (2048 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 50 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ± 1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 24 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | yes (selectable) |
| - general indication of wire break | | red LED |

| | | |
|---|---|---|
| Noise suppression for f = nx (50/60 Hz±1%) n = 1, 2, ... | | |
| - common mode rejection ($V_{pp}$=1 V) | min. | 86 dB |
| - series mode rejection (peak value of noise < rated value of input range) | min. | 40 dB |
| Basic error limits | | ± 0.15% |
| Operating error limits (0 to 60 °C) (32 to 140 °F) | | ± 0.4% |
| Single errors | | |
| - linearity | | ± 0.05% |
| - tolerance | | ± 0.05% |
| - polarity reversal error | | ± 0.05% |
| Temperature error | | |
| - final value | | ± 0.01 %/K |
| - zero point | | ± 0.002 %/K |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ | | none |
| Auxiliary current for PT 100 | | 2.5 mA |
| Resistance sensor | | |
| - tolerance | | ± 0.05% |
| - temperature error | | ± 0.006%/K |
| - load dependency | | ± 0.02%/100 |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to + 9 V) | | 60 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from + 9 V (CPU) | typ. | 70 mA |
| Power loss of the module | typ. | 0.9 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 2 x PT 100/±500 mV** **(6ES5 464-8MF21)**



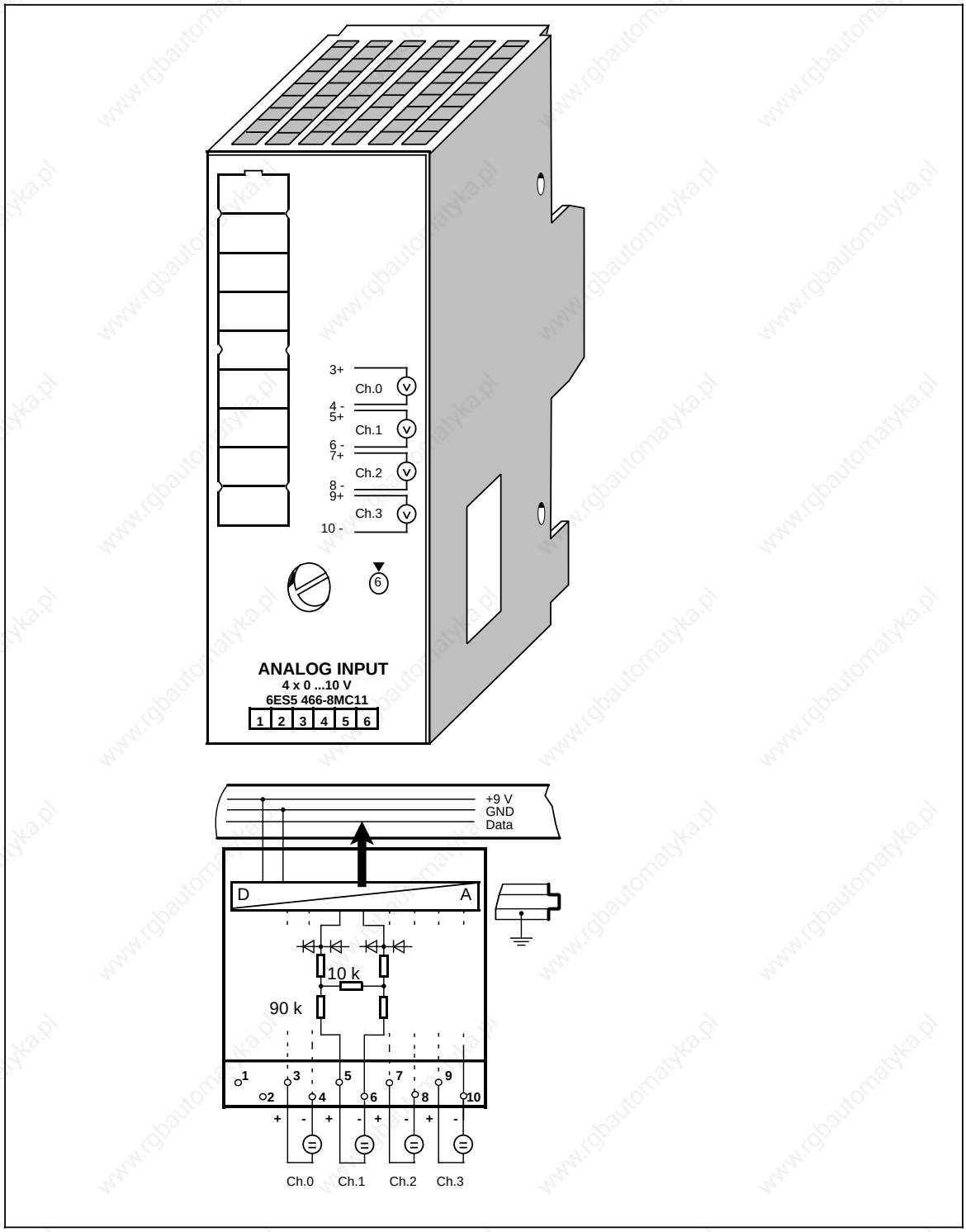
## Analog Input Module 2 × PT 100/±500 mV (continued) (6ES5 464-8MF21)

**Technical specifications**

| | | |
|---|---|---|
| Input range (rated values) | | |
| - resistance sensor (PT 100) | | 0 to 200 (max. 400 ) |
| - voltage source | | ± 500 mV |
| Number of inputs | | 1 or 2 (selectable) |
| Galvanic isolation | | yes (inputs to grounding point; not between inputs) |
| Input resistance | | 10 M |
| Connection method of sensors | | two or four-wire connection |
| Digital representation of input signal | | 12 bits + sign (2048 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Measuring principle | | integrating |
| Conversion principle | | voltage-time conversion (dual slope) |
| Integration time (adjustable for optimum noise suppression) | | 20 ms at 50 Hz<br>16.6 ms at 60 Hz |
| Encoding time per input | | |
| - for 2048 units | max. | 60 ms at 50 Hz |
| | max. | 50 ms at 60 Hz |
| - for 4095 units | max. | 80 ms at 50 Hz |
| | max. | 66.6 ms at 60 Hz |
| Permissible voltage difference | | |
| - between inputs | max. | ± 1 V |
| - between inputs and central ground point | max. | 75 V DC/60 V AC |
| Permissible input voltage (destruction limit) | max. | 24 V DC |
| Fault indication for | | |
| - range exceeded | | yes (more than 4095 units) |
| - sensor wire break | | yes (selectable) |
| - general indication of wire break | | red LED |

| | | |
|---|---|---|
| Noise suppression for f = nx (50/60Hz ± 1%); n = 1, 2, ... | | |
| - common mode rejection ($V_{PP}$ = 1 V) | min. | 86 dB |
| - series-mode rejection (peak value of noise < rated value of input range) | min. | 40 dB |
| Basic error limits | | ±0.15% |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.4 % |
| Single errors | | |
| - linearity | | ±0.05% |
| - tolerance | | ±0.05% |
| - polarity reversal error | | ±0.05% |
| Temperature error | | |
| - final value | | ±0.01%/K |
| - zero point | | ± 0.002%/K |
| Linearization exactness in rated range | | ±0.5 °C (0.9 °F) |
| Characteristic linearization of PT 100-characteristic curve | | DIN IEC 751 |
| Length of cable | | |
| - shielded | max. | 200 m (660 ft.) |
| Supply voltage L + | | none |
| Auxiliary current for PT 100 | | 2.5 mA |
| Resistance-type sensor | | |
| - tolerance | | ±0.05% |
| - temperature error | | ±0.006%/K |
| - influence of load variation | | ±0.02%/100 |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Rated insulation voltage (inputs to + 9V) | | 60 V AC |
| - insulation group | | 1 x B |
| - tested with | | 500 V AC |
| Current consumption | | |
| - from + 9 V (CPU) | typ. | 100 mA |
| Power loss of the module | typ. | 0.9 W |
| Weight | approx. | 230 g (8 oz.) |

**Analog Input Module 4 × +0 to 10 V** **(6ES5 466-8MC11)**



## Analog Input Module 4 × +0 to 10 V (continued) (6ES5 466-8MC11)

**Technical specifications**

| | | |
|---|---|---|
| Input ranges (rated values) | | +0 to 10 V |
| Number of inputs | | 4 |
| Galvanic isolation | | no |
| Input resistance | | 100 k |
| Connection for the signal sensor | | 2-wire connection |
| Digital representation of the input signal | | 8 bits (256 units = rated value) |
| Representation of the measured value | | binary * |
| Measuring principle | | successive approximation |
| Conversion time | | 100 µs |
| Encoding time per input | | 5 ms |
| Permissible voltage difference - between inputs | max. | ±1 V |
| Permissible input voltage (destruction limit) | max. | 60 V DC |
| Fault indication for - range exceeded | | no |
| - sensor wire break | | no |
| - general indication of wire break | | no |
| Noise suppression - common mode interference ($V_{PP}$=1 V) | min. | 86 dB |

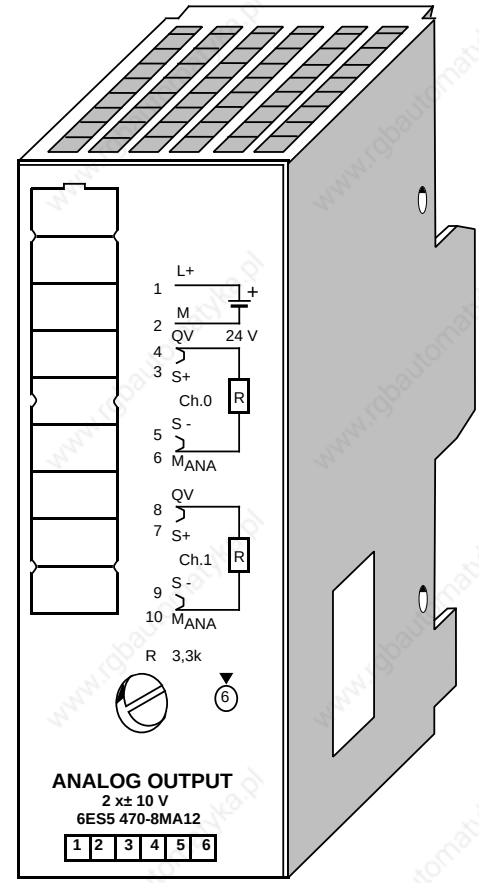
| | | |
|---|---|---|
| Basic error limits | | ±0.4% |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.6% |
| Single errors - linearity | | ±0.1% |
| - tolerance | | ±0.1% |
| Temperature error - final value | | ±0.01% K |
| - zero point | | ±0.01% K |
| Length of cable - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ | | none |
| Current consumption - from + 9 V (CPU) | typ. | 100 mA |
| Power loss of the module | typ. | 0.9 W |
| Weight | approx. | 200 g (7 oz.) |

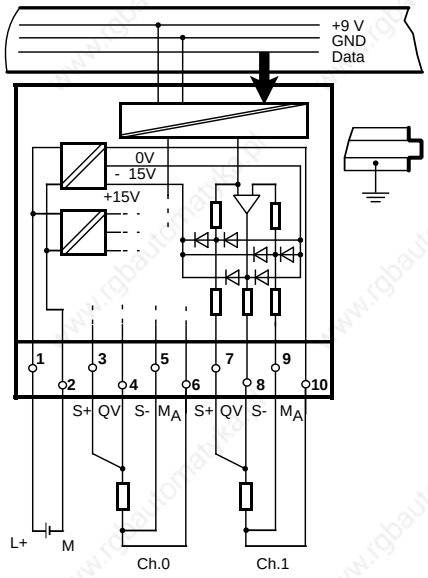
*

| Units | Input voltage in V | Bit 7 $2^7$ | 6 $2^6$ | 5 $2^5$ | 4 $2^4$ | 3 $2^3$ | 2 $2^2$ | 1 $2^1$ | 0 $2^0$ |
|---|---|---|---|---|---|---|---|---|---|
| 255 | 9.961 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 254 | 9.922 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 192 | 7.500 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 191 | 7.461 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 128 | 5.000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 127 | 4.961 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 64 | 2.500 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 63 | 2.461 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0.039 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0.000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 14.7.2 Analog Output Modules

### Analog Output Module 2 x ±10 V (6ES5 470-8MA12)



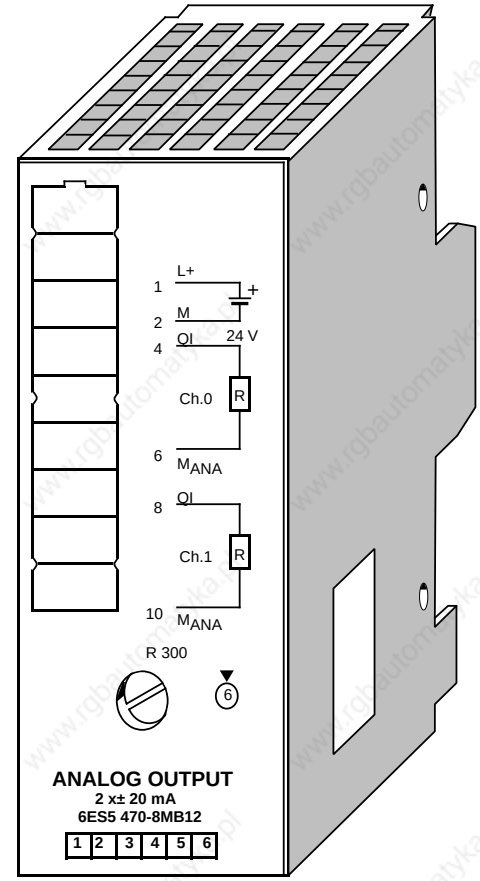



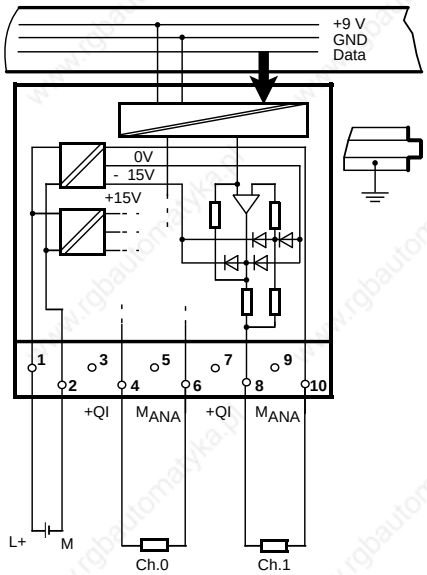

Legend:
QV: Analog output "voltage"

**Technical specifications**

| | | |
|---|---|---|
| Output range (rated values) | | ±10 V |
| Number of outputs | | 2 |
| Galvanic isolation | | yes (outputs to grounding point and between outputs) |
| Input resistance | min. | 3.3 k |
| Connection method | | two or four-wire connection |
| Digital representation of output signal | | 11 bits + sign (1024 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Conversion time (0 to 100%) | max. | 0.15 ms |
| Permissible overload | | 25% |
| Short-circuit protection | | yes |
| Short-circuit current | | ±30 mA |
| Permissible voltage difference to ground and between outputs | max. | 75 V DC/60 V AC |
| Basic error limits | | ±0.3% |
| Operational error limits (0 to 60 °C) (32 to 140 °F) | | ±0.6% |
| Single errors<br>- linearity<br>- polarity reversal error<br>- temperature error | | <br>±0.2%<br>±0.1%<br>±0.01%/K |
| Length of cable<br>- shielded | <br>max. | <br>200 m (660 ft.) |
| Supply voltage L+ (peripheral)<br>- rated value<br>- ripple $V_{PP}$<br>- permissible range (ripple included) | | <br>24 V DC<br>3.6 V<br>20 to 30 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ )<br>- insulation group<br>- tested with | | <br>12 V AC<br>1 x B<br>500 V AC |
| Rated insulation voltage (Output to L +, between outputs, output to + 9V)<br>- insulation group<br>- tested with | | <br>60 V AC<br>1 x B<br>500 V AC |
| Current consumption<br>- from + 9 V (CPU)<br>- from L + | <br>typ.<br>typ. | <br>120 mA<br>100 mA |
| Power loss of the module | typ. | 3.1 W |
| Weight | approx. | 220 g (8 oz.) |

## Analog Output Module 2 x ±20 mA (6ES5 470-8MB12)



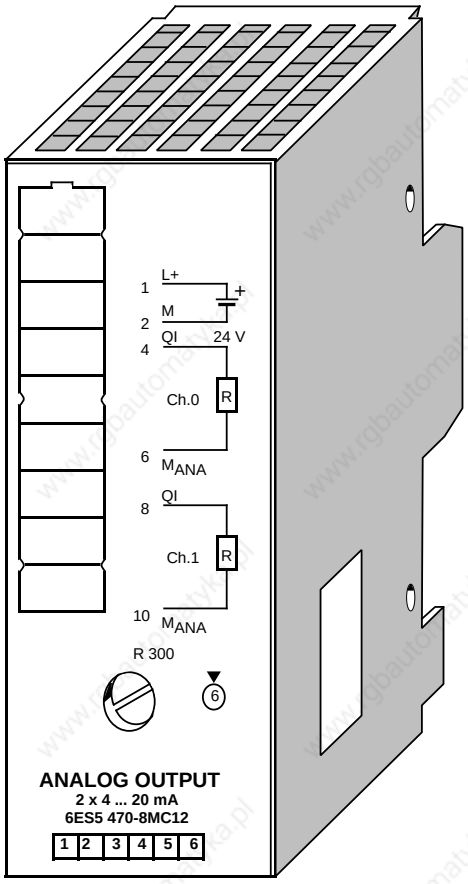



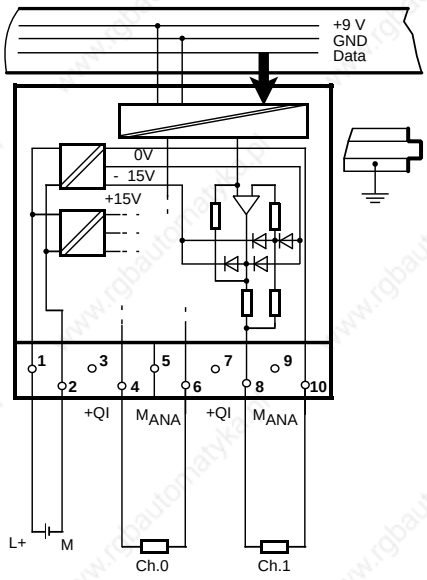

**Technical specifications**

| | | |
|---|---|---|
| Output range (rated values) | | ±20 mA |
| Number of outputs | | 2 |
| Galvanic isolation | | yes (outputs to grounding point and between outputs) |
| Input resistance | max. | 300 |
| Connection method | | two-wire connection |
| Digital representation of output signal | | 11 bits + sign (1024 units =rated value) |
| Measured value representation | | two's complement (left-justified) |
| Conversion time (0 to 100%) | max. | 0.15 ms |
| Permissible overload | | 25% |
| Short-circuit protection | | yes |
| Leerlaufspannung | | ± 12 V |
| Permissible voltage difference to central ground point and between outputs | max. | 75 V DC/60 V AC |
| Basic error limits | | ±0.3% |
| Operating error limits (0 to 60 °C) (32 to 140 °F) | | ±0.6% |
| Single errors - linearity - polarity reversal error - temperature error | | ±0.2% ±0.1% ±0.01%/K |
| Length of cable - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ - rated value - ripple $V_{PP}$ - permissible range (ripple included) | | 24 V DC 3.6 V 20 to 30 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) - insulation group - tested with | | 12 V AC 1 x B 500 V AC |
| Rated insulation voltage (output to L+, between outputs, output to + 9 V) - insulation group - tested with | | 60 V AC 1 x B 500 V AC |
| Current consumption - from + 9 V (CPU) - from L + | typ. typ. | 120 mA 130 mA |
| Power loss of the module | typ. | 3.8 W |
| Weight | approx. | 220 g (8 oz.) |

Legend:
QI: Analog output "current"

## Analog Output Module 2 x 4 to 20 mA (6ES5 470-8MC12)



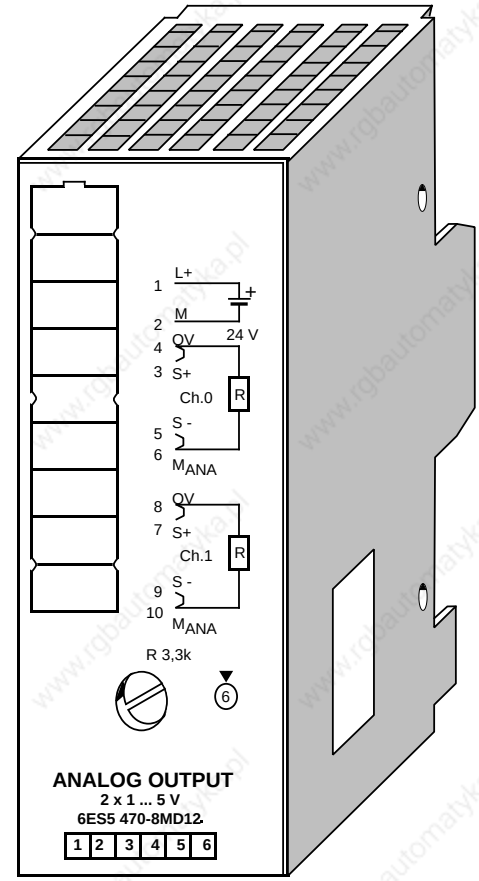



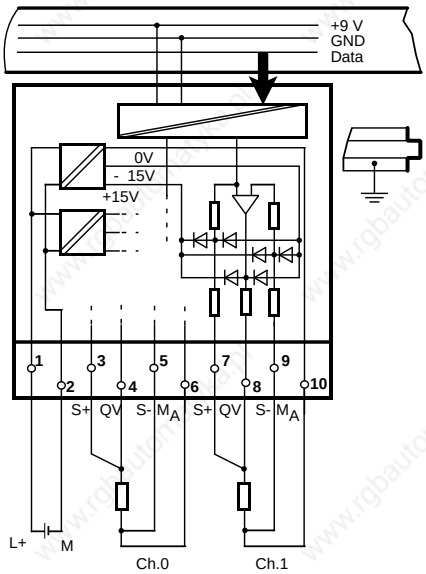

Legend:
QI: Analog output "current"

**Technical specifications**

| | | |
|---|---|---|
| Output range (rated value) | | 4 to 20 mA |
| Number of outputs | | 2 |
| Galvanic isolation | | yes (outputs to grounding point and between outputs) |
| Load resistance | max. | 300 |
| Connection method | | two-wire connection |
| Digital representation of output signal | | 11 bits + sign (1024 units = rated value) |
| Measured value representation | | two's complement (left-justified) |
| Conversion time (0 to 100%) | max. | 0.15 ms |
| Permissible overload | | 25% |
| Short-circuit protection | | yes |
| Leerlaufspannung | | ±12 V |
| Permissible voltage difference to central ground point and between outputs | max. | 75 V DC/60 V AC |
| Basic error limits | | ±0.2% |
| Operating error limits (0 to 60 °C) (32 to 140 °F) | | ±0.6% |
| Single errors - linearity - temperature error | | ±0.2% ±0.01%/K |
| Length of cable - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ - rated value - ripple $V_{PP}$ - permissible range (ripple included) | | 24 V DC 3.6 V 20 to 30 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚ ) - insulation group - tested with | | 12 V AC 1 x B 500 V AC |
| Rated insulation voltage (outputs to L+, between outputs, output to + 9 V ) - insulation group - tested with | | 60 V AC 1 x B 500 V AC |
| Current consumption - from + 9 V (CPU) - from L+ | typ. typ. | 120 mA 130 mA |
| Power loss of the module | typ. | 3.8 W |
| Weight | approx. | 220 g (8 oz.) |

## Analog Output Module 2 x 1 ... 5 V (6ES5 470-8MD12)




**Technical specifications**

| | | |
|---|---|---|
| Output range (rated values) | | 1 to 5 V |
| Number of outputs | | 2 |
| Galvanic isolation | | yes (outputs to grounding point and between outputs) |
| Input resistance | min. | 3.3 k |
| Connection method | | two or four-wire connection |
| Digital representation of output signal | | 11 bits + sign (1024 units=rated value) |
| Measured value representation | | two's complement (left-justified) |
| Conversion time (0 to 100%) | max. | 0.15 ms |
| Permissible overload | | 25% |
| Short-circuit protection | | yes |
| Short-circuit current | | ±30 mA |
| Permissible voltage difference to central ground point and between outputs | max. | 75 V DC/60 V AC |
| Basic error limits | | ±0.2% |
| Operating error limits (0 to 60 °C) (32 to 140 °F) | | ±0.6% |
| Single errors - linearity - temperature error | | ±0.2% ±0.01%/K |
| Length of cable - shielded | max. | 200 m (660 ft.) |
| Supply voltage L+ - rated value - ripple $V_{PP}$ - permissible range (ripple included) | | 24 V DC 3.6 V 20 to 30 V |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) - insulation group - tested with | | 12 V AC 1 x B 500 V AC |
| Rated insulation voltage (outputs to L+, between outputs, output to +9 V) - insulation group - tested with | | 60 V AC 1 x B 500 V AC |
| Current consumption - from +9 V (CPU) - from L+ | typ. typ. | 120 mA 100 mA |
| Power loss of the module | typ. | 3.1 W |
| Weight | approx. | 220 g (8 oz.) |

## 15 Function Modules


15.1 Comparator Module 2 × 1 to 20 mA/0.5 to 10 V . . . . . 15 - 1

15.2 Timer Module 2 × 0.3 to 300 s . . . . . 15 - 4

15.3 Simulator Module . . . . . 15 - 7

15.4 Diagnostic Module . . . . . 15 - 9

15.5 Counter Module 2 × 0 to 500 Hz . . . . . 15 - 12

15.6 Counter Module 25/500 kHz . . . . . 15 - 17
15.6.1 Installation Guidelines . . . . . 15 - 20
15.6.2 Data Transfer . . . . . 15 - 25
15.6.3 Functional Description of the Counter Mode . . . . . 15 - 27
15.6.4 Functional Description of the Position Decoder . . . . . 15 - 29
15.6.5 Entering New Setpoints for the Counter and Position Decoder . . . . . 15 - 38
15.6.6 Addressing . . . . . 15 - 39

15.7 Closed-Loop Control Module IP 262 . . . . . 15 - 41

15.8 IP 263 Positioning Module . . . . . 15 - 45

15.9 IP 264 Electronic Cam Controller Module . . . . . 15 - 49

15.10 IP 265 High Speed Sub Control . . . . . 15 - 52

15.11 Positioning Module IP 266 . . . . . 15 - 55

15.12 Stepper Motor Control Module IP 267 . . . . . 15 - 59

15.13 Communications Modules . . . . . 15 - 62
15.13.1 Printer Communications Module CP 521 . . . . . 15 - 62
15.13.2 Communications Module CP 521 BASIC . . . . . 15 - 65

## Figures

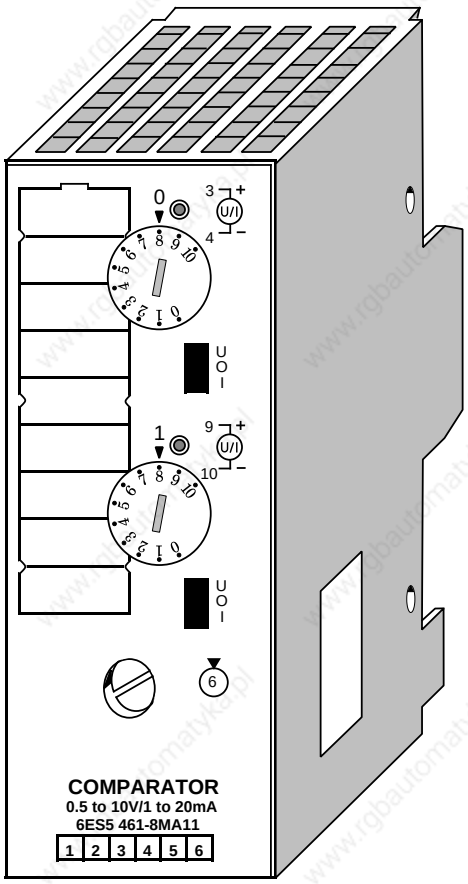

15-1 Scanning the Comparator Module . . . . . 15 - 2
15-2 Scanning the Timer Module . . . . . 15 - 5
15-3 Scanning the Simulator Module as a Digital Input . . . . . 15 - 8
15-4 Setting the Input Voltage Range on the Counter Module (500 Hz) . . . . . 15 - 14
15-5 Scanning the Counter Module (500 Hz) . . . . . 15 - 15
15-6 Timing Diagram: Setting and Resetting an Output of the Counter Module (500 Hz) . . . . . 15 - 15
15-7 Switch Positions on the Operating Mode Switch . . . . . 15 - 19
15-8 Pin Assignment of the 15-Pin Sub-D Female Connector . . . . . 15 - 20
15-9 Connecting a Counting Pulse Sensor for 5-V Differential Signal to RS 422 15 - 21
15-10 Connecting a Counting Pulse Sensor for 24 V DC . . . . . 15 - 21
15-11 Connecting a 5-V Position Sensor to RS 422 . . . . . 15 - 22
15-12 Connecting a 24-V DC Position Sensor . . . . . 15 - 22
15-13 Signal Sequence for Up-Counting . . . . . 15 - 23
15-14 Assignment Diagram for the Terminal Block . . . . . 15 - 24
15-15 Diagnostic Byte . . . . . 15 - 26
15-16 Switching the Outputs Dependent on the Status of the Counter and the Enable Input . . . . . 15 - 28
15-17 Position of the Reference Point (SYNC Bit 0 --> 1) within the Reference Signal Range . . . . . 15 - 32
15-18 Position of the Reference Point (SYNC Bit 0 --> 1) after the Reference Signal . . . . . 15 - 32
15-19a Synchronization (SYNC Bit 0 --> 1) during a Reversal of Direction before Reaching the Reference Pulse in a Positive Direction . . . . . 15 - 33
15-19b No Synchronization during a Reversal of Direction before Reaching the Reference Pulse in a Positive Direction . . . . . 15 - 33
15-20 Schematic of a Reference Point Approach Operation . . . . . 15 - 33
15-21 Enabling the Outputs - Reaching the Setpoints - Resetting the Outputs . . 15 - 34
15-22 Approaching a Setpoint in Up-Count Direction . . . . . 15 - 35
15-23 Approaching a Setpoint in Down-Count Direction . . . . . 15 - 36
15-24 Approaching a Setpoint in Up-Count Direction and Consecutive Reversal of Direction . . . . . 15 - 36
15-25 Requirement for New Setpoints . . . . . 15 - 38
15-26 Positioning with the IP 263 . . . . . 15 - 48
15-27 Units of Measurement that IP 266 Can Process for Circular Axis and Linear Axis . . . . . 15 - 56
15-28 Course of a Following Error during a Positioning Operation . . . . . 15 - 57
15-29 Velocity Profile of the IP 267 . . . . . 15 - 60

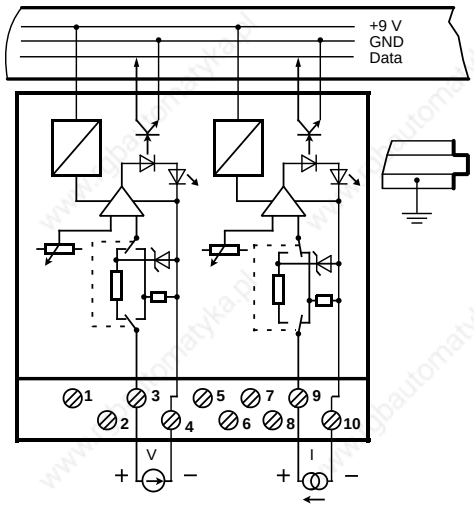

## Tables


15-1 Sending Data from the Programmable Controller to the Counter Module . . . 15 - 25
15-2 Sending Data from the Counter Module to the Programmable Controller . . 15 - 25
15-3 Pulse Evaluation . . . . . 15 - 30
15-4 Example for a Traversing Range . . . . . 15 - 31
15-5 Reaction of the Counter Module during Transfer of the Setpoints . . . . . 15 - 38
15-6 Slot Addressing . . . . . 15 - 39
15-7 Meaning of the Address Bytes of a Slot Address (Example: Slot 1) . . . . . 15 - 39
15-8 Designation of the Operating Modes . . . . . 15 - 58

# 15 Function Modules

## 15.1 Comparator Module 2 × 1 to 20 mA/0.5 to 10 V (6ES5 461-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| Channels | | 2 |
| Galvanic isolation | | yes |
| Current or voltage measurement | | switch-selectable |
| Switch position “0” | | no measuring |
| Display | | green LED for actual value setpoint |
| Setpoint adjustment | | with potentiometer |
| Setting error | | ±10% |
| Reproducibility | | ±2% |
| Hysteresis | | 10% |
| “V” measuring range | | 0.5 to 10 V DC |
| Input resistance | | 47 k |
| Inherent delay | typ. | 5 ms |
| Input voltage | max. | 100 V DC ( 0.5 s) |
| “I” measuring range | | 0.5 to 20 mA |
| Input resistance | | 500 |
| Overload capability | | 100% |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to measuring circuit and between measuring circuits) | | 30 V AC |
| - insulation | | 2×B |
| - tested with | | 500 V AC |
| Rated insulation voltage (+9 V to ⏚ ) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Length of cable | | |
| - shielded | | 200 m (660 ft.) |
| - unshielded | | 100 m (330 ft.) |
| Current consumption | | |
| - from +9 V (CPU) | typ. | 35 mA |
| Power losses of the module | typ. | 0.3 W |
| Weight | approx. | 200 g (7 oz.) |

### Function

The module has two isolated comparators for voltage or current measurement (selector switch with positions U/0/I). When the preset value is reached, the LED of the respective channel lights up and sends a “1” signal to the programmable controller.

The module must be removed or the measuring circuit disconnected before you select the function.

In switch position “0”, the comparator is switched off; if scanned, a “0” signal results.

The response threshold of the comparator is set by a selector on the front panel. The selector has scale divisions to simplify adjustment.

### Installation

The comparator module is mounted on a bus unit like any other input or output module (see chapter 3).

### Wiring

See schematic diagram. Unused inputs can be left open.

### Addressing

The comparator module is addressed like a 2-channel digital input module (channel “0” or “1”).

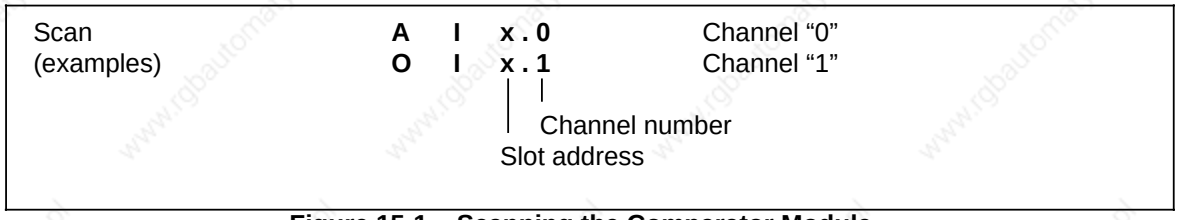
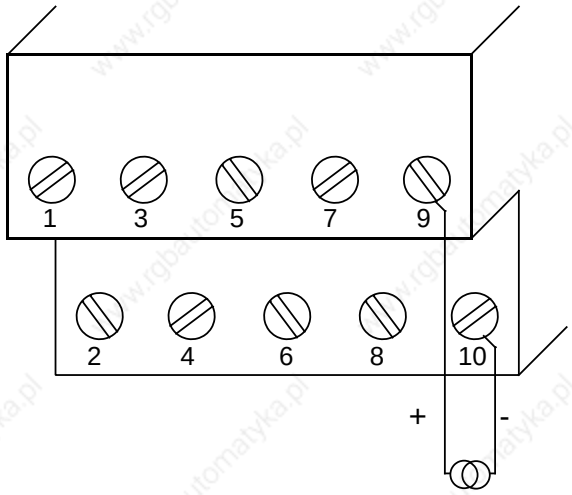



**Figure 15-1. Scanning the Comparator Module**

**Typical Application**

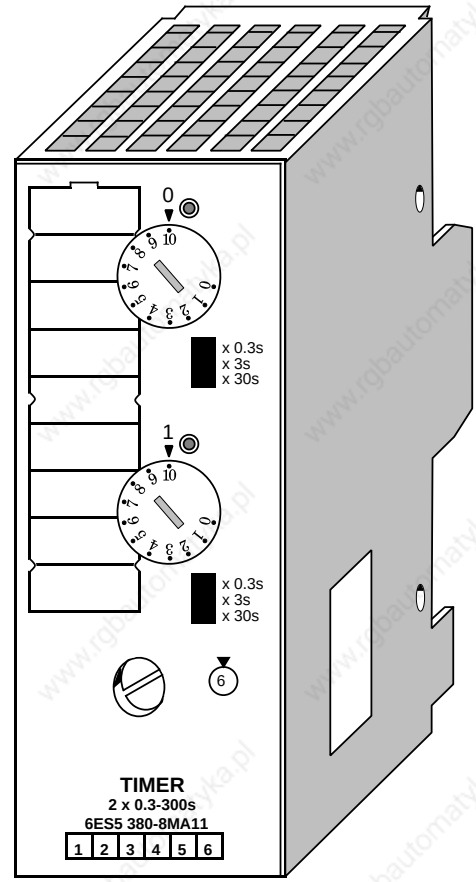
A comparator module is mounted at slot 4. The current source is connected to channel 1. If the Schmitt trigger 1 detects that the current has exceeded the preset value, output 5.1 is to be set.

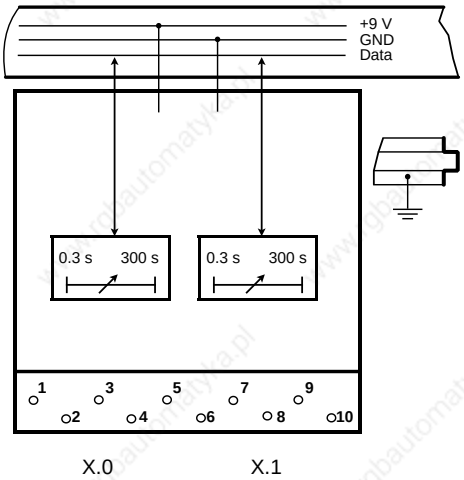
**Terminal Connections**




| STL | Explanation |
|---|---|
| A I 4.1<br>= Q 5.1 | As soon as the limit is reached or exceeded, input 4.1 becomes "1"; this sets output 5.1 to "1". |

## 15.2 Timer Module 2 × 0.3 to 300 s (6ES5 380-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| Number of timers | | 2 |
| Time setting<br>Range extension factor | | 0.3 to 3 s<br>×10, ×100 |
| Function indication | | green LED |
| Setting error | | ±10% |
| Reproducibility | | ±3% |
| Temperature influence | | +1%/10 °C (50 °F) of set time |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage<br>(+ 9 V to ⏚ )<br>- insulation group<br>- tested with | | <br>12 V AC<br>1×B<br>500 V AC |
| Current consumption<br>- from +9 V (CPU) | <br>typ. | <br>10 mA |
| Weight | approx. | 200 g (7 oz.) |

### Function

The module contains two pulse timers. While a timer runs, the LED of the respective channel is lit and a “1” is reported to the CPU.

The pulse duration is preselected with the time range selector “x 0.3 s / x 3 s / x 30 s” in a definite range and then set to the exact value by means of a potentiometer on the front panel. This selector has scale divisions to simplify setting.
(time value=time range x scale value)

| | | |
|---|---|---|
| Example: | Time range: | x 3 s |
| | Scale value: | 7 |
| | Set time: | 7 x 3 s=21 s |

### Installation

The counter is inserted into a bus unit like any other input or output module (see chapter 3).

### Wiring

No wiring is required.

### Addressing

A timer module is addressed like a two-channel digital module (channel “0” or “1”).

The timer module is addressed like a digital output module for starting, resetting, or interrupting the pulse. The signal status is scanned like a digital input module.

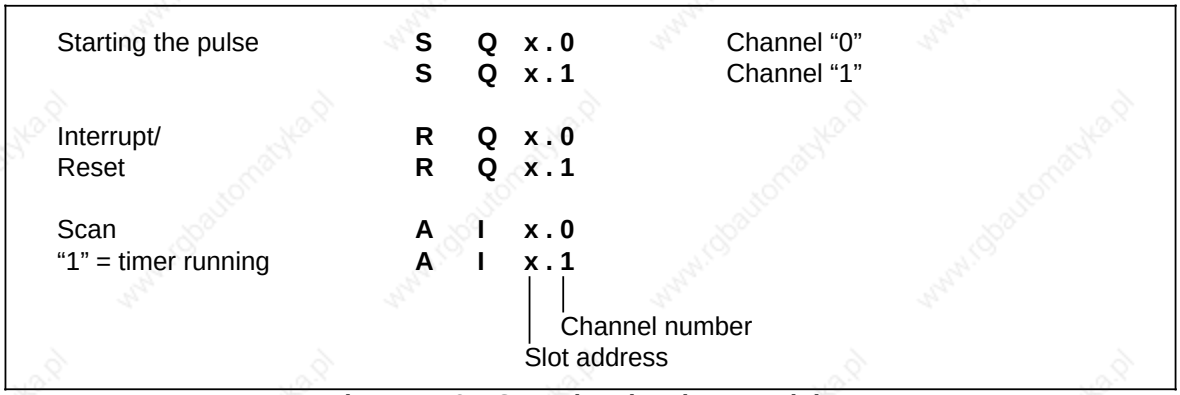


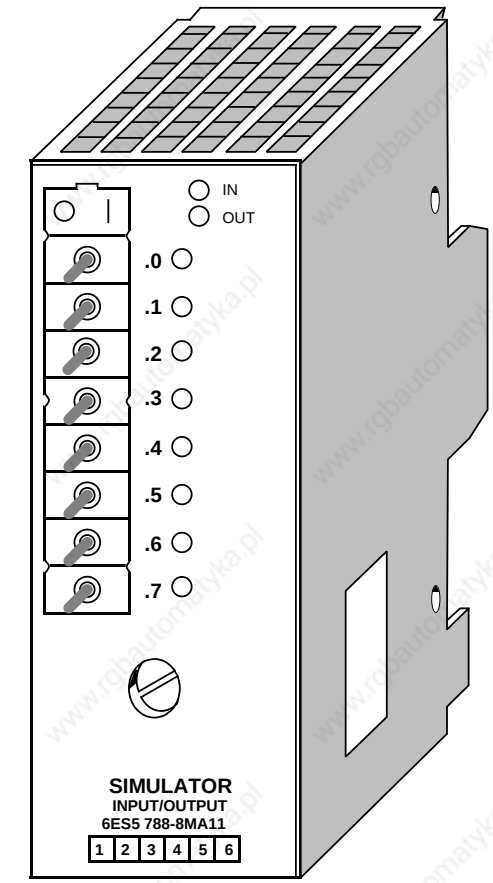

**Figure 15-2. Scanning the Timer Module**

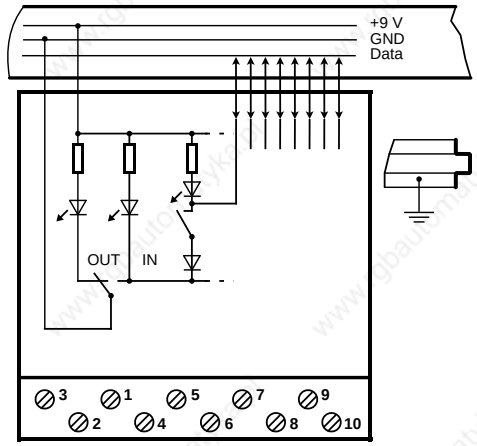
**Typical Application as** “**On-Delay Timer**”

A timer module is mounted at slot 5. A time of 270 s is set on channel “0” of this module by means of the time-range selector and the potentiometer. The timer is started when input 0.0 is “1”.
A lamp lights up (output 4.0) when the timer has run down.

| **Terminal Connections** | |
|---|---|
| No process peripherals are connected to this module.<br>Unlike the internal timers, times can be set or modified using a timer module without making any program modifications. | |
| **STL** | **Explanation** |
| A I 0.0<br>AN I 5.0<br>A F 65.0<br>S Q 4.0<br>A I 5.0<br>= F 65.0<br>AN I 0.0<br>R Q 4.0<br>A I 0.0<br>= Q 5.0 | The timer must not be scanned in the program scan cycle in which it was enabled since the CPU would not receive the acknowledgement that the timer had started until one program scan later.<br>If flag 65.0 is “1” and the timer has run down (AN I 5.0), output 4.0 is set to “1”.<br>If the “Timer started” message has been sent to the CPU, the flag is set.<br><br>If I 0.0 is “0”, the lamp is switched off.<br>The timer is started if I 0.0 is “1”. |

## 15.3 Simulator Module (6ES5 788-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| Function selection<br>- simulation of 8 input signals<br>- display of 8 output signals | | selected by switch on rear of module |
| Function indication | | yellow LED |
| "0"/"1" input signals | | switch-selectable |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage<br>(+9 V to ⏚)<br>- insulation group<br>- tested with | | <br>12 V AC<br>1×B<br>500 V AC |
| Signal status display for input/output | | green LEDs |
| Current consumption<br>- from +9 V (CPU) | | <br>30 mA |
| Power loss of the module | typ. | 0.3 W |
| Weight | | 190 g (6.7 oz.) |

### Function

Simulator modules are 8-channel modules that can simulate digital input signals and display output signals.

The type of module to be simulated (input or output) is selected by means of a switch on the rear of the module and indicated by two LEDs on the front panel.

The module cannot simulate interrupt inputs.

### Installation

The simulator module is inserted into a bus unit like any other input or output module (see chapter 3). The module does not have a coding key and can therefore replace any digital module. The coding element on the bus unit does not have to be readjusted.

### Wiring

There is no electrical connection between the module and the terminal block. It can therefore be inserted into slots that have already been wired and connected to the power supply.

### Addressing

A simulator module is addressed like an 8-channel digital module (channels 0 to 7).

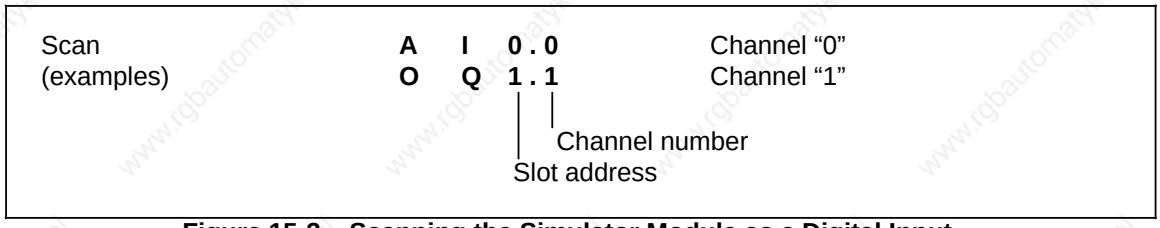



**Figure 15-3. Scanning the Simulator Module as a Digital Input**

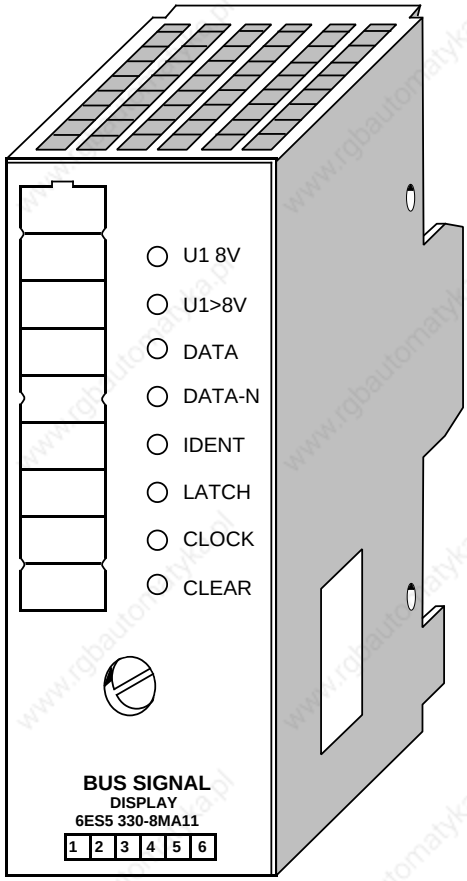
### Typical Application

The CPU is in the RUN mode, the green LED is lit but the programmable controller is malfunctioning. You have also discovered that the fault must be in a particular I/O module. If the module has no fault indicator, check to see if:

- The power supply is connected.
- The bus connections and interface modules are plugged in correctly.

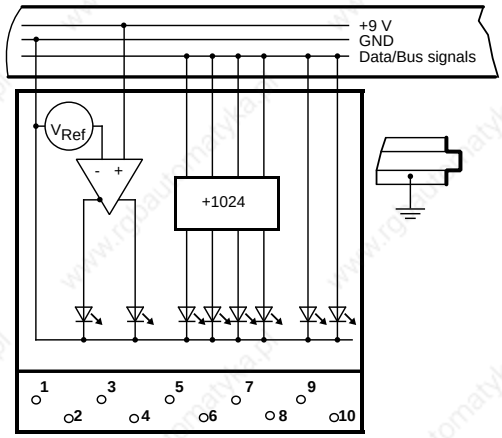
Then, try to access the module via the process image (STATUS or STATUS VAR). If this procedure is not successful, replace the module with the simulator module. Perform a second check with the STATUS or STATUS VAR function. If the simulator performs, the input/output module you replaced is defective.

## 15.4 Diagnostic Module (6ES5 330-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (+9 V to ⏚) | | 12 V AC |
| - insulation group | | 1×B |
| - tested with | | 500 V AC |
| Voltage monitor | | |
| - undervoltage | | red LED |
| - voltage ok | | green LED |
| Signal status display for control signals | | yellow LEDs |
| Current consumption - from+9 V (CPU) | | 25 mA |
| Power loss of the module | typ. | 0.3 W |
| Weight | approx. | 175 g (6.1 oz.) |



**Function**

The diagnostic module is used for monitoring the S5-100U I/O bus. LEDs on the front panel display the signal states of the control lines and the supply voltage for the I/O bus.

- IDENT
  The programmable controller executes an IDENT run after each change from STOP to RUN. It executes an IDENT run after any changes in the configuration in order to determine the current configuration. The IDENT LED lights up briefly. If the LED lights up in the RUN mode, this indicates that a faulty I/O module has been plugged in.
- CLEAR
  The CLEAR signal line is “1” only in the STOP mode in normal operation.
  The outputs of the output modules are disabled.
  If CLEAR is “1” in the RUN mode, the control line itself may be defective (no contact).
- LATCH/CLOCK
  These two control lines control data interchange between the CPU, the I/O bus, and the I/O modules.
  During normal operation, both LEDs must flash (programmable controller in RUN mode).
  The flashing frequency provides information on the speed of the serial bus.
  If both LEDs show a steady light in the RUN mode, the bus unit that the diagnostic module is plugged into is defective.
- DATA/DATA-N
  The alternate lighting up of the DATA and DATA-N LEDs indicates data flow on the I/O bus.
  If these two LEDs show a steady light (as in the case of the LATCH and CLOCK LEDs), this indicates that the bus unit that the diagnostic module is plugged into is defective.
- U1 8 V
  If the supply voltage of a slot remains at a value U1 8 V, proper functioning of the I/O modules is no longer guaranteed. The low supply voltage can be explained by an excessively high bus load (> 1 A).
  If this LED flickers, noise pulses are superimposed on the supply voltage U1 (e.g., by the coupling of noise pulses).
  The LED lights up briefly if the programmable controller is switched on or off.
- U1 > 8 V
  The supply voltage of the I/O bus is functioning correctly.

### Installation

The diagnostic module is plugged into a bus unit like any other input or output module (see chapter 3). The module has no mechanical coding. The coding element on the bus unit does not have to be reset.

> **Note**
>
> The module can be plugged in and removed regardless of the operating status of the programmable controller.

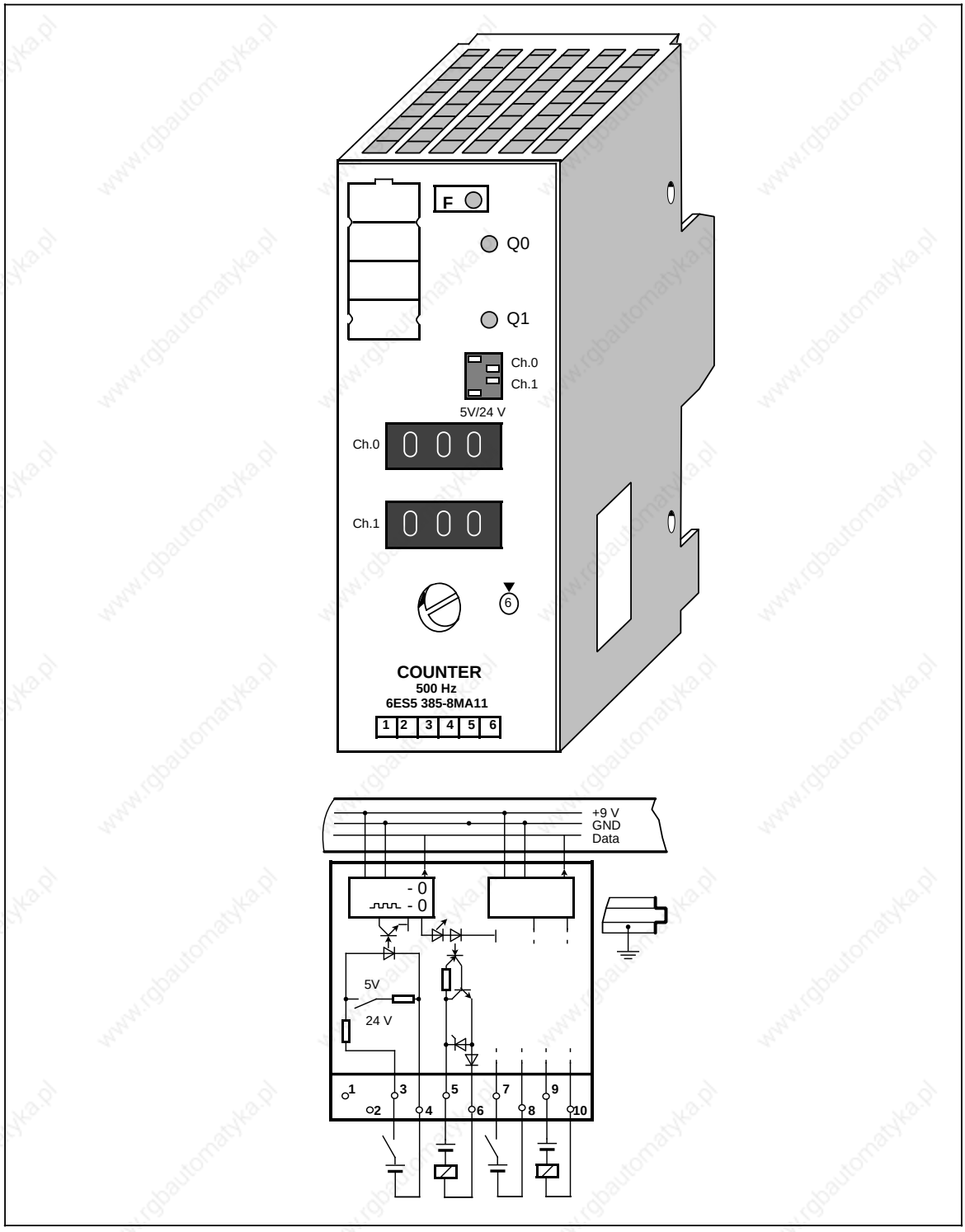
### Wiring

No wiring is required. Existing wiring does not have to be removed.

### Addressing

There is no addressing since the module cannot be addressed by the programmable controller.

## 15.5 Counter Module 2 × 0 to 500 Hz (6ES5 385-8MA11)



**Technical Specifications**

| | | |
|---|---|---|
| **Number of Inputs** | | 2 |
| Galvanic isolation | | yes |
| Input voltage | | |
| - rated value | | 5 V/24 V DC |
| - for “0” signal | | 0 to 0.8/-33 to 5 V |
| - for “1” signal | | 3 to 5 V/13 to 33 V |
| Input current for “1” signal | typ. | 1.5/8.5 mA |
| Inherent delay | typ. | 180 µs |
| Input frequency | max. | 500 Hz |
| Connection of 2-wire BERO proximity switches (24 V DC) | | possible |
| - quiescent current | | 1.5 mA |
| Length of cable | | |
| - unshielded | max. | 50 m (165 ft.) |
| **Number of Outputs** | | 2 |
| Galvanic isolation | | yes |
| Supply voltage L+ (for load) | | |
| - rated value | | 24 V DC |
| - permissible range (including ripple) | | 20 to 30 V |
| Output current for “1” signal | | |
| - rated value | | 0.5 A |
| - permissible range | | 5 to 500 mA |
| - lamp load | max. | 5 W |
| Residual current at “0” signal | max. | 1 mA |
| Output voltage | | |
| - for “0” signal | max. | 3 V |
| - for “1” signal | max. | L+-2.5 V |
| Short-circuit protection | | electronic |
| Fault indication (red LED) | | short-circuit |
| Voltage induced on circuit interruption (internal) limited to | | L+-47 V |
| Switching frequency | | |
| - resistive load | max. | 100 Hz |
| - inductive load | max. | 2 Hz |

| | | |
|---|---|---|
| Total permissible current of outputs | | 1 A |
| Driving a digital input | | possible |
| Paralleling of outputs | | possible |
| - max. current | | 0.5 A |
| Permissible ambient temperature for the unit | | |
| - horizontal arrangement | | 0 to 60 °C (32 to 140 °F) |
| - vertical arrangement | | 0 to 40 °C (32 to 104 °F) |
| Length of cable | | |
| - unshielded | max. | 100 m (330 ft.) |
| Insulation rating | | VDE 0160 |
| Rated insulation voltage (inputs and outputs to each other and to ⏚ ; input to +9 V) | | 60 V AC |
| - insulation group | | 1×B |
| - tested with | | 1250 V AC |
| Current consumption | | |
| - from+9 V (CPU) | typ. | 20 mA |
| Power loss of the module | typ. | 2.5 W |
| Weight | approx. | 200 g (7 oz.) |

### Function

The module consists of two independent down counters with isolated inputs and outputs. It counts input signals up to a frequency of 500 Hz from a set value down to the value 0. When 0 is reached, the 24-V DC output of the module is energized.

At the same time, a green LED on the module lights up and the input signal (I x.0 or I x.1) is set to “1”.

The setpoint (0 to 999) can be entered via the three-digit thumbwheel switches on the front panel of the module.

The input voltage ranges can be set for 5 V DC or 24 V DC using rocker switches on the front panel.

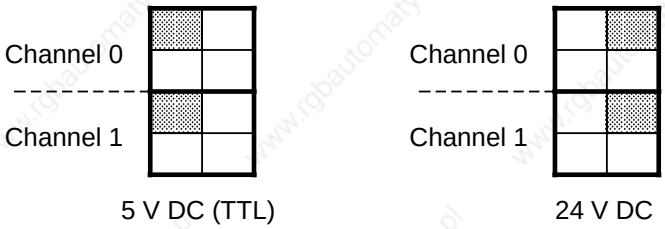



**Figure 15-4. Setting the Input Voltage Range on the Counter Module (500 Hz)**

### Installation

The counter module is plugged into a bus unit like any other module (see chapter 3).

### Wiring

See schematic diagram for the counter module.

### Addressing

A counter module can be addressed like a two-channel digital module (channel “0” or “1”). For enabling and resetting the counter, you address the module like a digital output module. The counter reading is scanned in the same way as a digital input module.

| | | | | |
|---|---|---|---|---|
| Counter enable | **S** | **Q** | **x . 0** | Channel “0” |
| (Set to start value) | **S** | **Q** | **x . 1** | Channel “1” |
| Counter reset | **R** | **Q** | **x . 0** | |
| | **R** | **Q** | **x . 1** | |
| Scan | **A** | **I** | **x . 0** | |
| “1” = Counter at zero | **A** | **I** | **x . 1** | |

Channel number
Slot address

**Figure 15-5. Scanning the Counter Module (500 Hz)**

### Timing Diagram

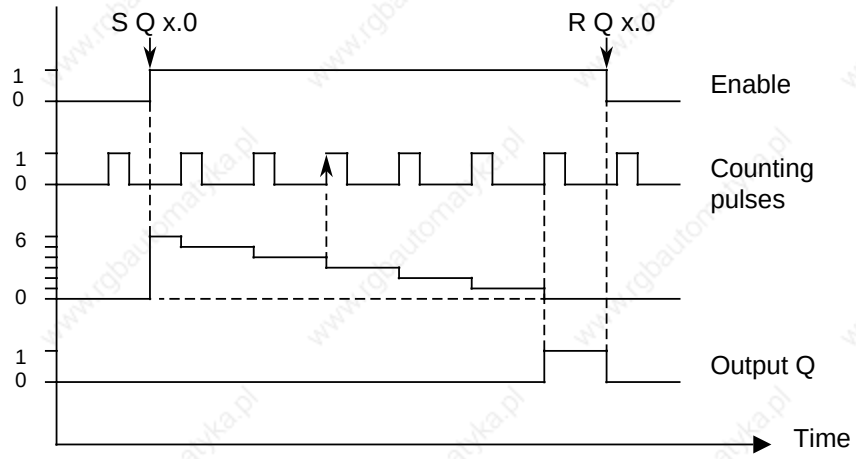
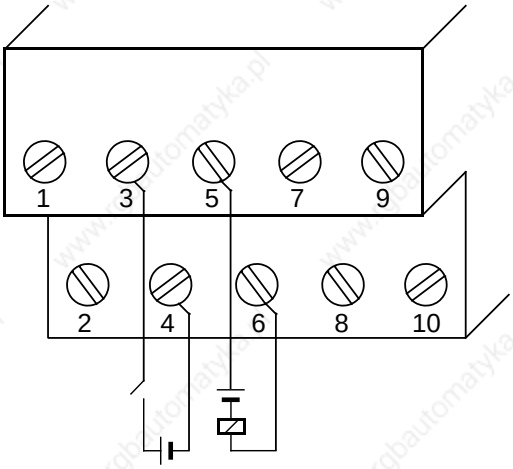



**Figure 15-6. Timing Diagram: Setting and Resetting an Output of the Counter Module (500 Hz)**

**Typical Application**

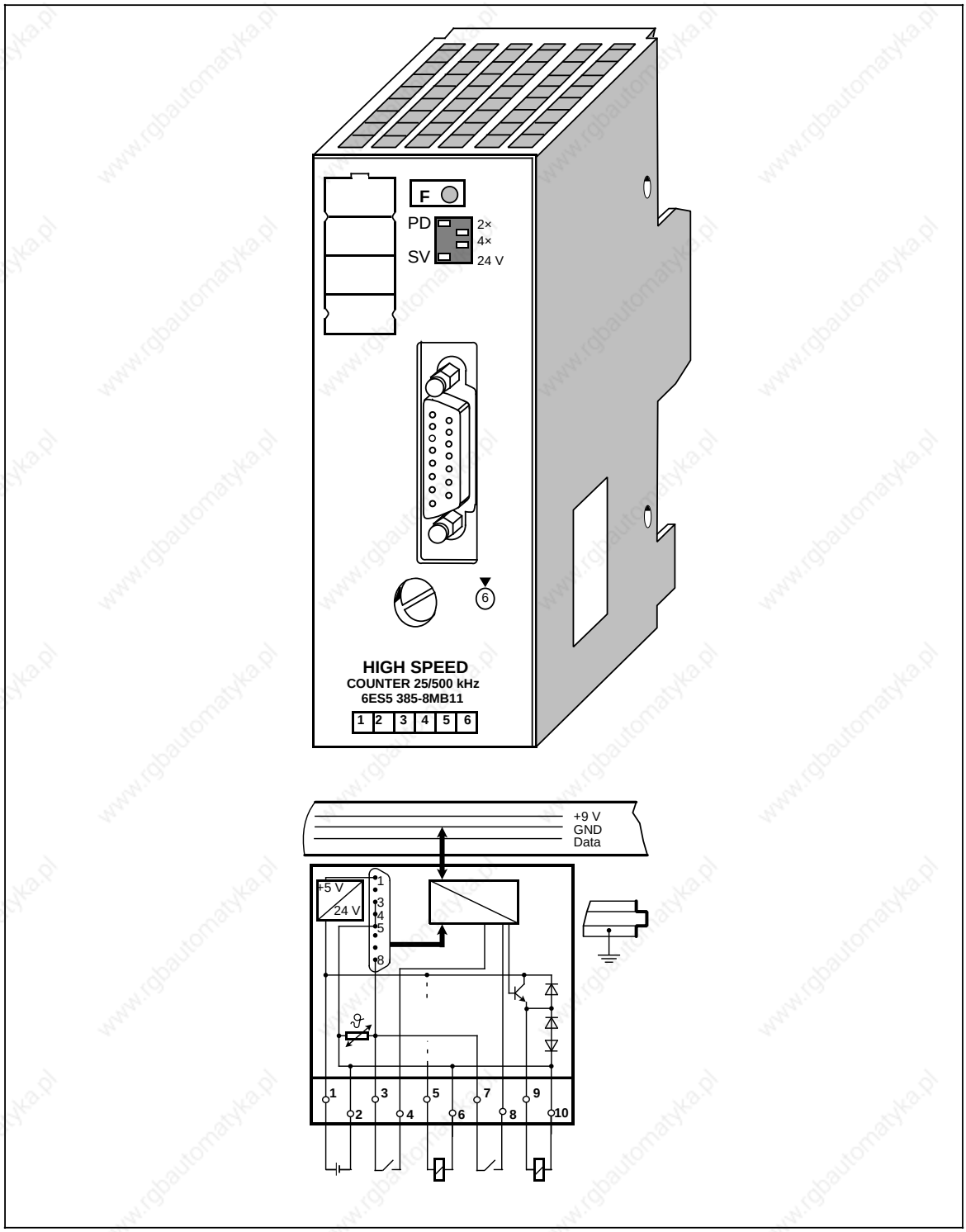
A counter module is plugged into slot 2. A value of 100 is set on channel “0” of this module using the three-digit thumbwheel switches. The incoming pulses are counted once the counter has been enabled by the control program. As soon as 100 pulses have been counted, a signal (output 4.0) is released.

**Connection Diagram**




| STL | Explanation |
|---|---|
| S Q 2.0<br>A I 2.0<br>= Q 4.0 | During the counting operation, input I 2.0 is “0”. When the counter status is “0”, input I 2.0 is set to “1”. Output Q 4.0 is then also set. Direct output at terminals 5 and 6 is possible. |

## 15.6 Counter Module 25/500 kHz (6ES5 385-8MB11)



**Technical Specifications**

| | | |
|---|---|---|
| Operating mode (switch-selectable) | | |
| - position decoder | | PD |
| - counter | | C |
| Sensor inputs | | 1 sensor 5 V (differential input) or 1 sensor 24 V DC |
| Digital inputs | | 2; reference and enabling |
| Digital outputs | | 2; setpoints reached 1 and 2 |
| Galvanic isolation | | no |
| Counting range | | |
| Operating mode | | |
| - position decoder | | two's complement (KF) - 32768 to +32767 |
| - counter | | unipolar representation (KH) 0 to 65535 |
| Counting mode | | |
| - position decoder | | up/down |
| - counter | | up |
| Setpoint input | | via program |
| **5-V Sensor Input** | | 15-pin Cannon sub-miniature D connector |
| Input signals | | differential signals to RS 422 |
| - position decoder | | A A-N, B B-N, R R-N |
| - counter | | A A-N |
| Counting frequency | max. | 500 kHz |
| Cable length (shielded) | max. | 50 m (165 ft.) |
| Power supply for decoder | | 5 V from L+via voltage transformer |
| Output current | max. | 300 mA, short-circuit-proof |
| **24-V Sensor Input** | | 15-pin Cannon sub-miniature D connector |
| Rated input voltage | | 24 V DC |
| Input signals | | |
| - position decoder | | A, B, R |
| - counter | | A |
| Input voltage | | |
| - "0" signal | | - 33 to 5 V DC |
| - "1" signal | | +13 to 33 V DC |
| Rated input current for "1" signal | typ. | 8.5 mA |
| Counting frequency | max. | 25 kHz |
| Cable length (shielded) | max. | 100 m (330 ft.) |
| Power supply for sensor | | 24 V from L+ (PTC thermistor) |
| Output current | max. | 300 mA, short-circuit proof |
| **Digital Inputs** | | reference and enabling |
| Rated input voltage | | 24 V DC |
| Input voltage | | |
| - "0" signal | | - 33 to +5 V DC |
| - "1" signal | | +13 to 33 V DC |
| Rated input current for "1" signal and at 24 V | typ. | 8.5 mA |
| Input frequency | max. | 100 Hz |
| Inherent delay | typ. | 3 ms (1.4 to 5 ms) |
| Cable length (unshielded) | max. | 100 m (330 ft.) |
| **Digital Outputs** | | setpoints 1 and 2 |
| Output current (resistive, inductive load) | | 5 mA to 0.5 A |
| Residual current for "0" signal | max. | 0.5 mA |
| Switching current for lamps | | 0.22 A (5 W) |
| Limitation of inductive interrupting voltage | to | -15 V |
| Output voltage | | |
| - "1" signal | min. | L+ - 2.2 V |
| - "0" signal | max. | 3 V |
| Cable length (unshielded) | max. | 100 m (330 ft.) |
| Short-circuit protection (cable impedance up to 15 ) | | electronic |
| Short-circuit indication (short-circuit to M) | | red LED |
| Supply voltage L+ | | |
| - rated value | | 24 V DC |
| - ripple $V_{pp}$ | max. | 3.6 V |
| - permissible range (including ripple) | | 20 to 30 V DC |
| Fuse (internal) | | T 5 A (slow blow) |
| Current consumption | | |
| - from L+ without sensor supply without load | | 30 mA |
| - internal (+9 V) | | 70 mA |
| Power consumption of the module | typ. | 1.9 W+total output current ($I_A$)×1.1 V |
| Weight | approx. | 250 g (9 oz.) |

**Function**

The counter module can be used as an up-counter or as an up/down counter for a position decoder. The counting pulses are supplied by a sensor that you can connect to the 15-pin subminiature D female connector of the module. You can choose from two types of sensors that fulfill the following requirements:

- 5-V error voltages according to RS 422 (up to 500 kHz)
- 24-V signals (up to 25 kHz)

As additional inputs, the module has an enable input and a reference input.

By using the STEP 5 program, you can assign two setpoints via the I/O bus. Once the counter status reaches one of these values, the respective output completes the circuit at terminal block (Q 0 or Q 1). The status of the outputs is displayed in the diagnostic byte.

You can also read the following values by using the STEP 5 program:

- The updated count
- The diagnostic byte

You can preselect the following items on the operating mode switch:

- The function mode
- The position resolution
- The input voltage range of the sensor

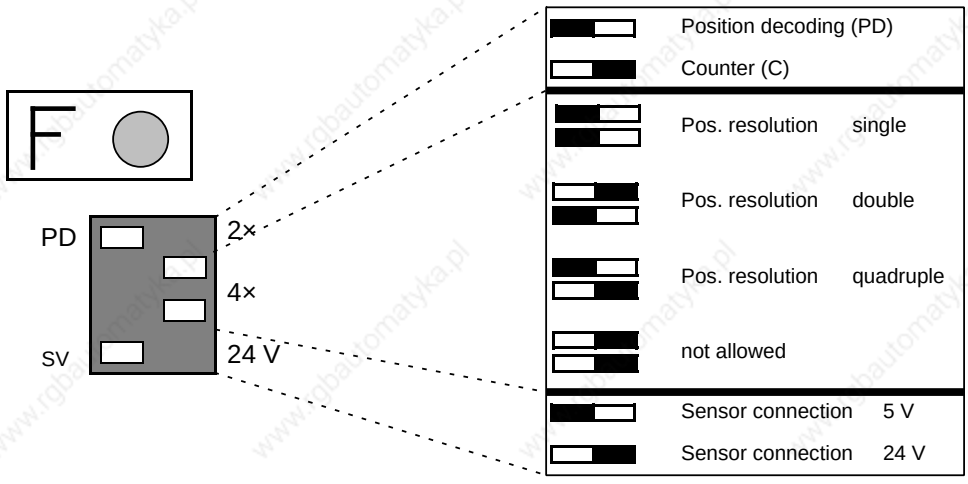



**Figure 15-7. Switch Positions on the Operating Mode Switch**

## 15.6.1 Installation Guidelines

### Installing and Removing the Module

Plug the counter module into a bus unit like other I/Os.
The counter module can only be plugged into slots 0 through 7.
Set the coding key to number 6 on the bus unit.

### Installing or Removing the Sensor

Disconnect the 24-V DC power supply (terminals 1 and 2 of the terminal block) before connecting or disconnecting the sensor cables.

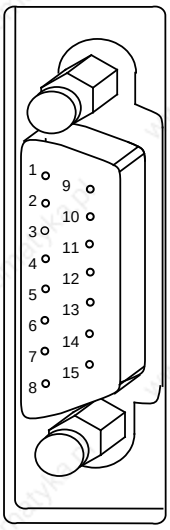
> **Warning**
>
> Connecting or disconnecting the 5-V sensor cable while the module is energized can cause damage to the sensor.

### Connection of Pulse and Position Sensors

Connect pulse and position sensors on the front plate by means of a 15-pin sub-D female connector. The correct cable connectors are listed in Appendix D. The module can supply the sensors (5 V DC or 24 V DC).

Basically, all sensors can be connected if they fulfill the requirements of the system signals and supply voltage. Sensors with OPEN-COLLECTOR outputs cannot be connected to the module.

The shield connection of the sensors must be connected to the metallic front connector cover.




| Pin | Assignment |
|---|---|
| 1 | 5 V Supply voltage |
| 2 | 5 V Sensor line |
| 3 | Ground |
| 4 | Ground |
| 5 | Ground |
| 6 | Rectangular-wave signal A-N (5 V) |
| 7 | Rectangular-wave signal A (5 V) |
| 8 | Supply voltage (24 V) |
| 9 | Rectangular-wave signal B (5 V) |
| 10 | Rectangular-wave signal B-N (5 V) |
| 11 | Reference pulse R (5 V) |
| 12 | Reference pulse R-N (5 V) |
| 13 | Rectangular-wave signal A (24 V) |
| 14 | Rectangular-wave signal B (24 V) |
| 15 | Reference pulse R (24 V) |

**Figure 15-8. Pin Assignment of the 15-Pin Sub-D Female Connector**

- **Connecting Counting Pulse Sensors for 5-V Differential Signal to RS 422**

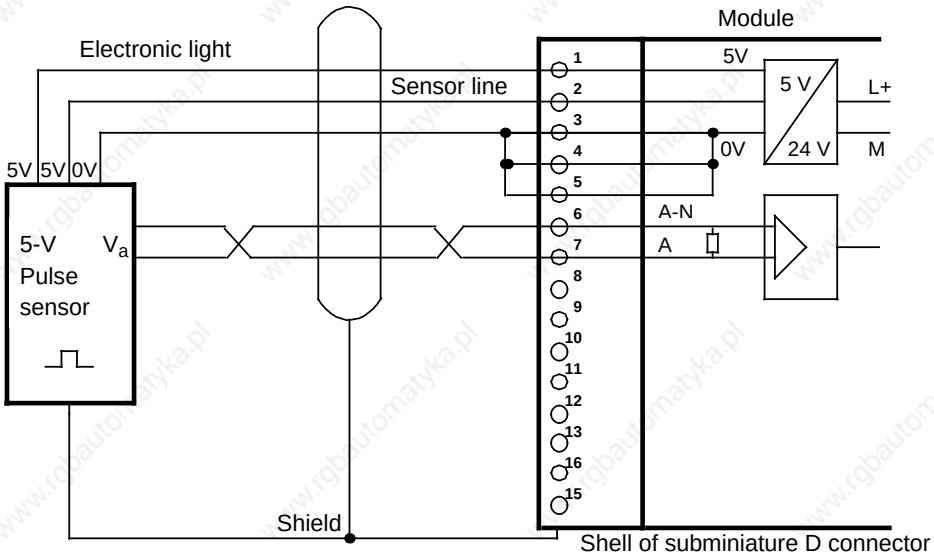



**Figure 15-9. Connecting a Counting Pulse Sensor for 5-V Differential Signal to RS 422**

- **Connecting a Counting Pulse Sensor for 24 V DC**

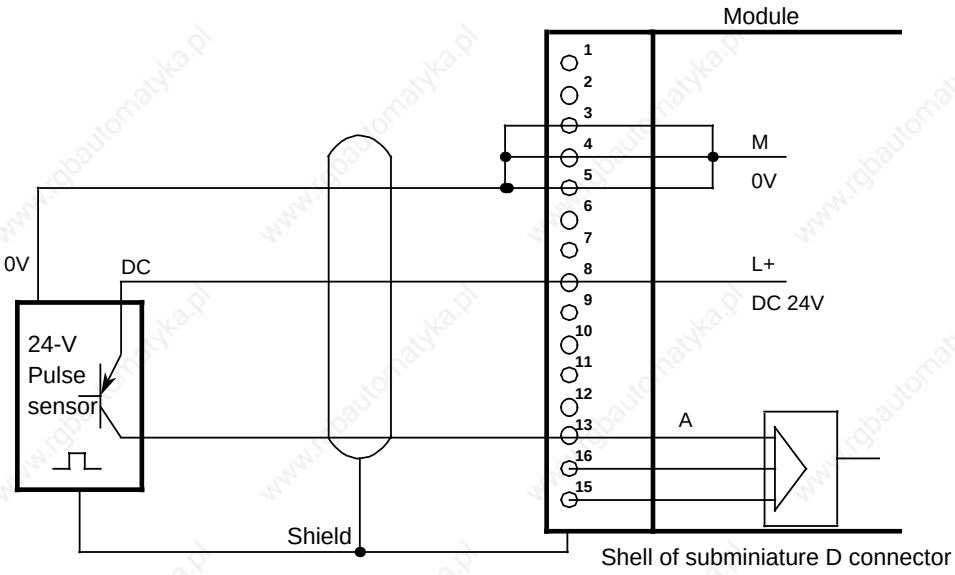



**Figure 15-10. Connecting a Counting Pulse Sensor for 24 V DC**

- **Connecting a 5-V Position Sensor to RS 422**

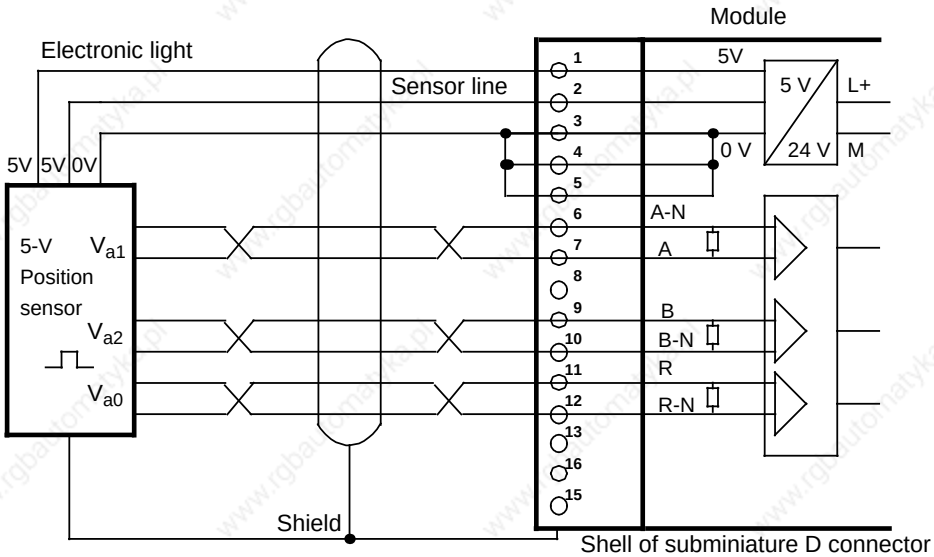



**Figure 15-11. Connecting a 5-V Position Sensor to RS 422**

- **Connecting a 24-V DC Position Sensor**

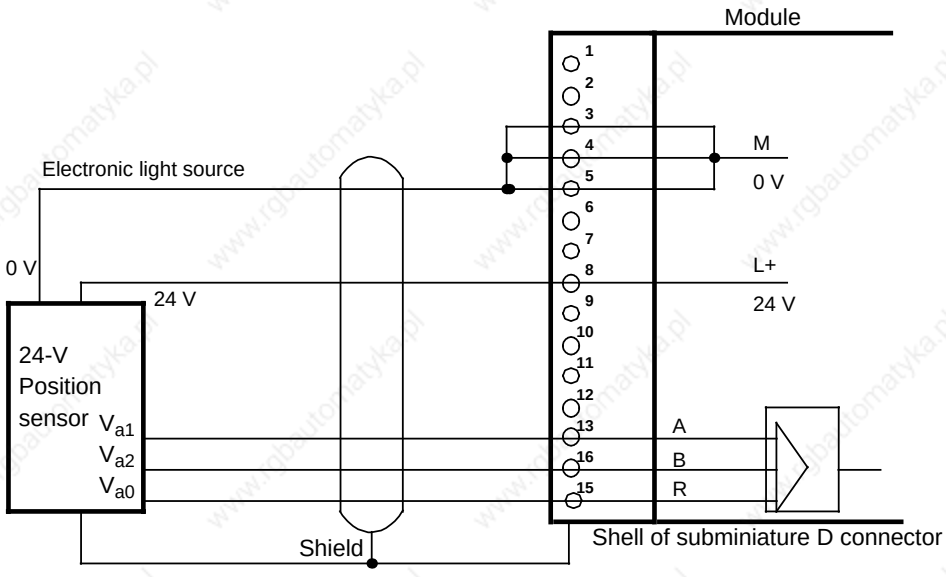



**Figure 15-12. Connecting a 24 V DC Position Sensor**

**Sensor Requirements**

The following requirements must be satisfied by the sensor signals to the module inputs:

- Signal sequence for up-counting

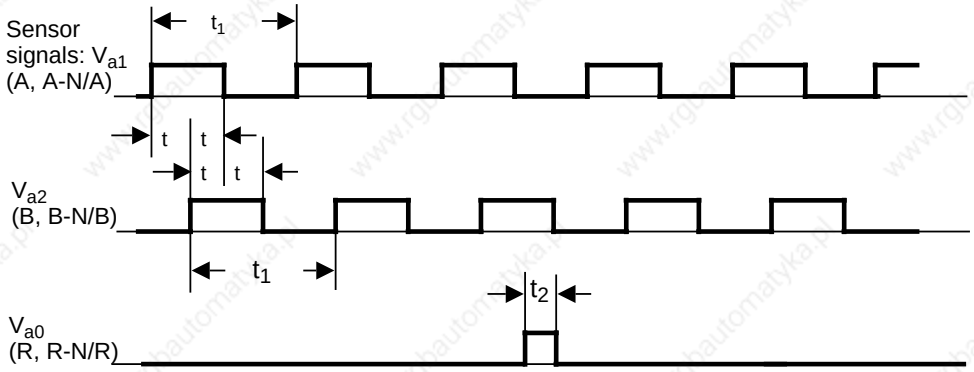



**Figure 15-13. Signal Sequence for Up-Counting**

- Pulse time of the sensors

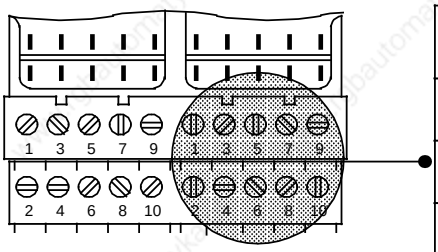
| | 5-V Sensors | 24-V Sensors | Pulses |
|---|---|---|---|
| t | 500 ns | 10 µs | $V_{a1}$ = Position decoder count pulses (A) |
| $t_1$ | 2 µs | 40 µs | $V_{a2}$ = Position decoder count pulses (B) |
| $t_2$ | 500 ns | 10 µs | $V_{a0}$ = Position decoder ref. pulse (R) |

- Minimum edge steepness

| | | | |
|---|---|---|---|
| 5 V | - | differential signals according to RS 422A (A, A-N, B, B-N, R, R-N): | 5 V/µs |
| 24 V | - | count pulses and reference pulse (A, B, R): | 0.3 V/µs |
| 24 V | - | enable and reference signal: | 0.3 mV/µs |

**Terminal Block**

Proximity switches can be connected (contacts, two-wire BERO proximity limit switches) to the inputs on the terminal block.




| Terminal | Terminal Assignment |
|---|---|
| 1<br>2 | 24-V DC supply for the module<br>Ground |
| 3<br>4 | 24-V DC supply for enable signal<br>DI enable signal |
| 5<br>6 | DQ 24 V/0.5 A setpoint (Q0)<br>Ground |
| 7<br>8 | 24-V DC supply for reference signal<br>DI reference signal |
| 9<br>10 | DQ 24-V/0.5 A setpoint 2 (Q1)<br>Ground |

**Figure 15-14. Assignment Diagram for the Terminal Block**

- **Assignment of Inputs on the Terminal Block**

  Two-wire BERO proximity limit switches can be connected to the reference input. The enable input can also be driven by a 24-V DC digital output module.

- **Outputs on the Terminal Block**

  There are two short-circuit protected 24-V DC digital outputs on the terminal block.

- **Short-Circuit Indication**

  A shorted output is indicated by the red LED on the front panel.

## 15.6.2 Data Transfer

The data is transmitted via the I/O bus. Four bytes are used.
Examples of data transfer are shown in section 15.6.6.

**Transfer from the Programmable Controller to the Counter Module (PIQ)**

The control program transfers two setpoints to the counter module by means of transfer operations.

**Table 15-1. Sending Data from the Programmable Controller to the Counter Module**

| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|
| Setpoint 1 | | Setpoint 2 | |
| High byte | Low byte | High byte | Low byte |

**Transfer from the Counter Module (PII) to the Programmable Controller**

The counter module transfers the diagnostic byte and the current counter status. In the control program, this data can be read in by means of load operations and then evaluated.

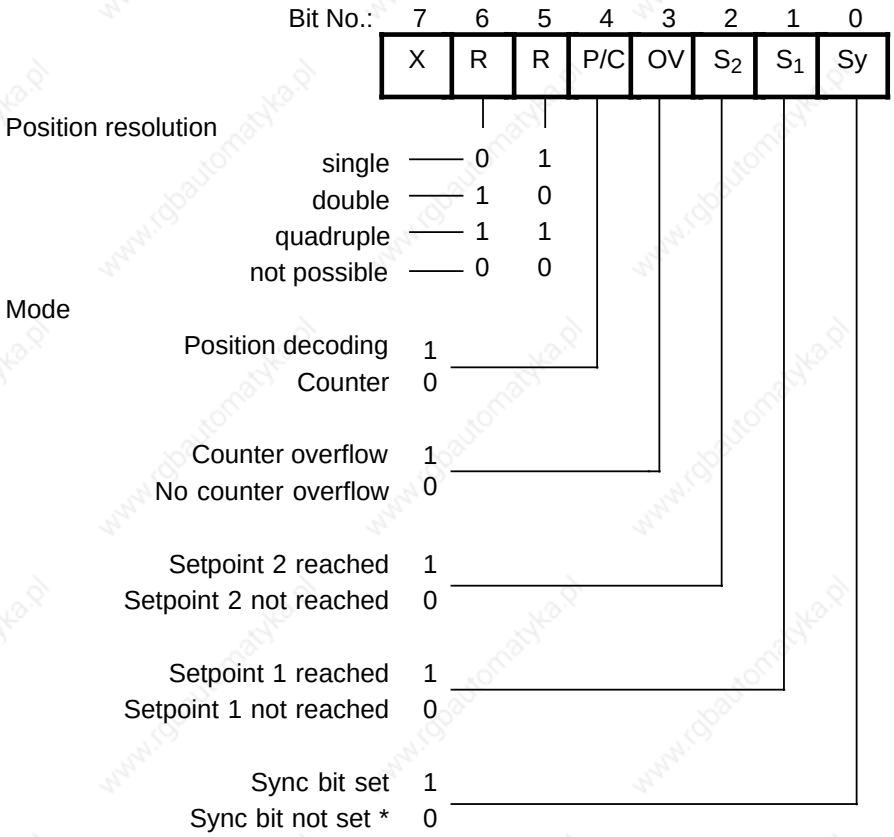
**Table 15-2. Sending Data from the Counter Module to the Programmable Controller**

| Byte 0 | Byte 1 | Byte 2 | Byte 3 |
|---|---|---|---|
| Irrelevant | **Diagnostic byte** | Actual value | |
| | | High byte | Low byte |

- **Diagnostic Byte (Byte 1)**

  The diagnostic byte is byte 1 of the first input word. Byte 0 has no significance.

  The diagnostic byte provides information on the following items:
  - Preset position resolution
  - Preset mode
  - Status of setpoints
  - Signal status of the sync bit for position decoding




X = irrelevant

***** If the sync bit is not set, a reference point approach must be implemented before operation can continue in the Position Decoding mode.

**Figure 15-15. Diagnostic Byte**

## 15.6.3 Functional Description of the Counter Mode

In the operation mode “Counter”, the module works as a “port-controlled” up-counter and counts the positive edges of the counting pulses while the enable input is active. If the counter reaches a preselected setpoint, the respective output is enabled.

**Initial Settings**

Use the operating mode switch to make the following selections:

- “Counter” (C)
- Signal level of counting pulses (5 V or 24 V)

The position of the switches for the position resolution is irrelevant.

For this operation, you need a counting pulse sensor (e.g., BERO). The pulses can be applied as 5-V differential signals according to RS 422A (up to 500 kHz) or as 24-V signals (up to 25 kHz). The sensor is connected to the sub-D connector of the module.

**Loading Setpoints**

The control program can transfer two setpoints to the module. These setpoints must be between 0 and 65,535.

The transfer of the setpoints via the module depends on whether the “setpoint 1 (setpoint 2) reached” bit is set in the diagnostic byte (S1 and S2).

If the bit is not set, which means the existing setpoint has not been reached or has not been exceeded, the new setpoint is transferred immediately and is immediately valid.

If the bit is set, which means the existing setpoint has been reached or exceeded, the new setpoint is valid only after a positive edge occurs at the enable input.

If you do not specify a setpoint, a setpoint of “0” applies.

**Enabling the Counter**

The signal state of the enable input (terminal 3 on the terminal block) determines the function of the counter.

A **positive edge** at the enable input

- Sets the counter to 0
- Resets the diagnostic bits for “setpoint reached”
- Resets the outputs
- Enables the counter

> **Note**
>
> The enable input should be set to “1” only after the setpoint has been transferred. Otherwise, the outputs are enabled automatically when the first positive edge occurs.

### Disabling the Counter

A **negative edge** at the enable input disables the counter. The outputs, diagnostic bits, and the counter are not reset. You can continue reading the current count. A positive edge at the enable input resets the outputs and the diagnostic bytes.

### Reaching the Setpoints - Setting the Outputs - Resetting the Outputs

If setpoints have been preselected and the counter is enabled, the module counts the positive edges at the counter input. The count is incremented by “1” with every leading edge.

After setpoint 1 has been reached, output Q 0 is enabled. At the same time, status bit S1 is set. After setpoint 2 has been reached, output Q 1 is enabled. At the same time, status bit S2 is set.

As long as the enable input is active, the counter counts the pulses. After the enable command has been cancelled, the counter is disabled. The actual value remains constant.

You can read the current count in the STEP 5 program. The actual value is displayed as an unsigned whole number and must be between 0 and 65,535.

> **Note**
>
> If no setpoint is preselected, the respective value “0” is assigned. The corresponding output is enabled with the positive edge of the enable input.

**Example:** Setpoints S1=2 and S2=4 are entered into the counter

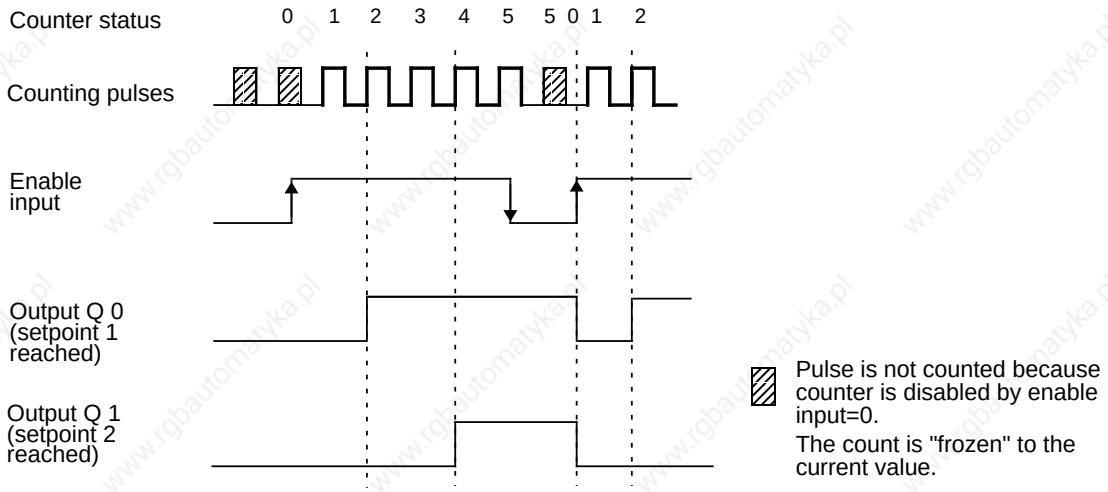



**Figure 15-16. Switching the Outputs Dependent on the Status of the Counter and the Enable Input**

When the programmable controller goes from RUN to STOP, outputs Q 0 and Q 1 are reset.

**Performance during Overflow**

If the enabled counter exceeds the counter range limit 65,535 the following occurs:

- Bit 3 (overflow) in the diagnostic byte is set to “1”.
- The outputs and diagnostic bits for “setpoint reached” are disabled, but they remain unchanged.

The counting function continues. Thus the actual value is constantly updated.

You can continue to read all data from the module in the STEP 5 program:

- The updated count
- The status of the outputs at the time of the overflow (This status remains unchanged until the overflow bit is reset.)
- The set overflow bit

After an overflow, the counter can be reset by one of the following actions:

- A positive edge at the enable input
- An overall reset of the programmable controller (STOP to RUN mode)

> **Note**
>
> After a cold restart of the programmable controller, the outputs are disabled. These outputs can be enabled via a positive edge to the enable input.

## 15.6.4 Functional Description of the Position Decoder

In the operation mode “position decoder” the module works as an up/down-counter and counts the pulses of the connected position encoder. Because of the phase offset of the two decoder signals A and B, the counter determines the counting direction. If the counter reaches a preselected set-point, the respective output is then turned on.

**Settings**

Set the following items on the operating mode switch:

- “Position decoding” (PD) function
- The desired position resolution (single, double, or quadruple)
- The signal level of the counting pulses (5 V or 24 V)

Connect the sub-D interface female connector to an incremental position encoder that has to deliver the following signals:

- Two counting pulses offset by 90 degrees
- A reference pulse

The pulses can be supplied as 5-V differential signals according to RS 422 (up to 500 kHz) or as 24-V DC signals.

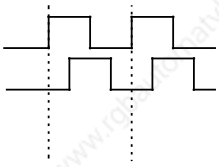
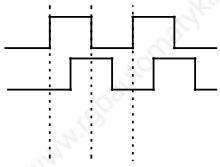
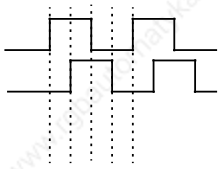
Connect a switch to the enable input. This switch must deliver a 24-V signal. In the same way, the reference pulse has to deliver a 24-V signal to the reference input.

**Position Resolution**

- Counter capacity
  The 16-bit up/down-counter permits a resolution of 65,536 units between -32,768 and +32,767. The traversing range depends on the resolution of the position encoders.
- Pulse evaluation
  The counting pulses, which are offset by 90 degrees, can be subjected to single, double, or quadruple evaluation. The necessary setting is made on the operating mode switch (see section 15.6).

The accuracy of the traversing path increases accordingly if double or quadruple pulse evaluation is used. However, the traversing range then available is reduced by the factor 2 or 4.

**Table 15-3. Pulse Evaluation**

| | Single Evaluation | Double Evaluation | Quadruple Evaluation |
|---|---|---|---|
| Counting pulse A<br>Counting pulse B |  |  |  |
| Count | 0 1 | 0 1 2 | 0 1 2 3 4 |

**Example:**

A rotary incremental position encoder produces 1000 pulses per revolution.

The spindle has a pitch of 50 mm/revolution. The position encoder therefore produces 1000 pulses for a traversing path of 50 mm (1 revolution).

The resolution of the encoder is therefore 50 mm/1000 pulses.

The counter can handle up to 65,536 pulses. With the above resolution, the following traversing ranges are obtained:

**Table 15-4. Example for a Traversing Range**

| Pulse evaluation | Single | Double | Quadruple |
|---|---|---|---|
| Traversing range | 3.25 m (10.7 ft.) | 1.625 m (5.3 ft.) | 0.81 m (2.7 ft.) |
| Distance travelled/ pulse | 50 µm | 25 µm | 12.5 µm |

**Loading Setpoints**

In the STEP 5 program, two setpoints can be transferred to the module. These setpoints must lie between -32768 and +32767.

The acceptance of the setpoints by the module depends on whether the “setpoint 1 (setpoint 2) reached” bit has been set in the diagnostic byte.

If the bit is not set, which means the existing setpoint is not reached or not exceeded, the new setpoint is immediately accepted and is immediately valid.

If the bit is set, which means the existing setpoint is reached or exceeded, the new setpoint is not valid until a positive edge occurs at the enable input.

If you do not specify a setpoint, a setpoint of “0” applies.

**Synchronization of the Actual Value Detection (Reference Point Approach)**

The synchronization of the actual value detection is necessary after Power ON and after a counter overflow.

Synchronization performs one of the following functions:

- The count (actual value) is set to “0” and the **SYNC bit** (bit 0 in the diagnostic bit) is **set** after Power ON.
- The **overflow bit** (bit 3 in the diagnostic byte) is **reset** after an overflow.

**Prerequisites for a Synchronization**

1. The reference signal

   The sensor for the reference signal is connected to terminals 7 and 8 of the terminal block.

   Synchronization is enabled with the **leading edge** (0 to 1) at terminal 8. If the signal was already on “1” when the module was switched on, then the reference signal must be turned off to restart the synchronization.
   If the reference signal lies in the normal traversing range, the actual value will be constantly resynchronized by the reference signal. To prevent the unwanted resynchronizing, you have to mask out the reference signal after the first reference point approach.

2. The traversing direction after a positive edge of the reference signal

   After the reference signal has been reached, the module has to recognize a **positive traversing path** (up-counting) while the reference signal is still active (1). This means, you have to input the reference signal with increasing actual value to synchronize the module.

3. The reference pulse

   The reference pulse is generated by the position encoder at least once per revolution.
   - The first reference pulse that the module recognizes after a leading edge of the reference signal synchronizes the module (see figures 15.17 and 15.19a).
   - If the reference signal changes from ”1” to ”0” before the reference pulse is reached, the module is only synchronized if a positive traversing path is recognized after the falling edge of the reference signal (see figure 15.18).
     The module is not synchronized, if a negative traversing path is recognized after the falling edge of the reference signal (see figure 15.19b).

   The figures 15.17, 15.18 and 15.19a illustrate different possibilities for a reference traversing path. Figure 15.19b illustrates a reference traversing path, which is terminated without synchronization:

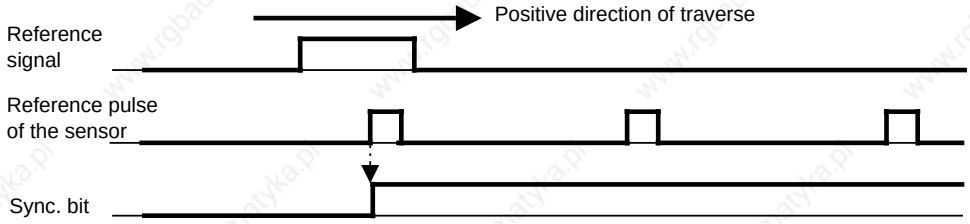



**Figure 15-17. Position of the Reference Point (SYNC Bit 0 --> 1) within the Reference Signal Range**

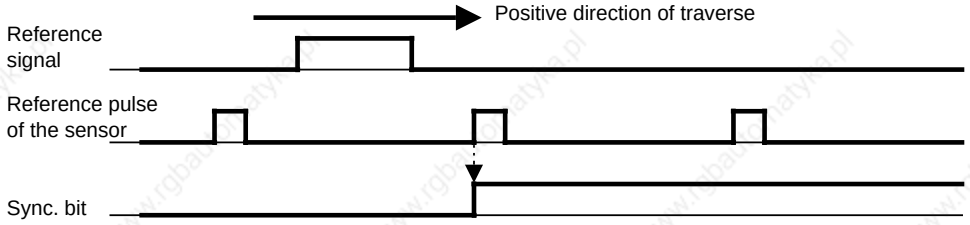



**Figure 15-18. Position of the Reference Point (SYNC Bit 0 --> 1) after the Reference Signal**

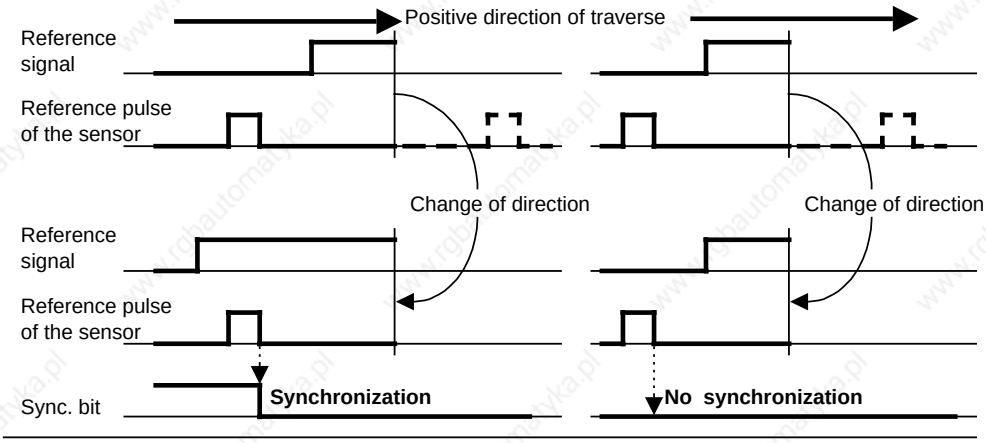



**Figure 15-19a. Synchronization (SYNC Bit 0 -->1) 15-19b. No Synchronization during a Reversal of Direction before Reaching the Reference Pulse in a Positive Direction**

**Example:** Transporting objects from point A to point B on a conveyor belt.

A rotary position encoder is used, together with a BERO proximity switch as reference transmitter. The conveyor belt is marked at a definite point. As soon as this mark comes within the range of the BERO, the BERO produces a reference signal.

Following the reference point approach, the enable input is set via a digital output module.

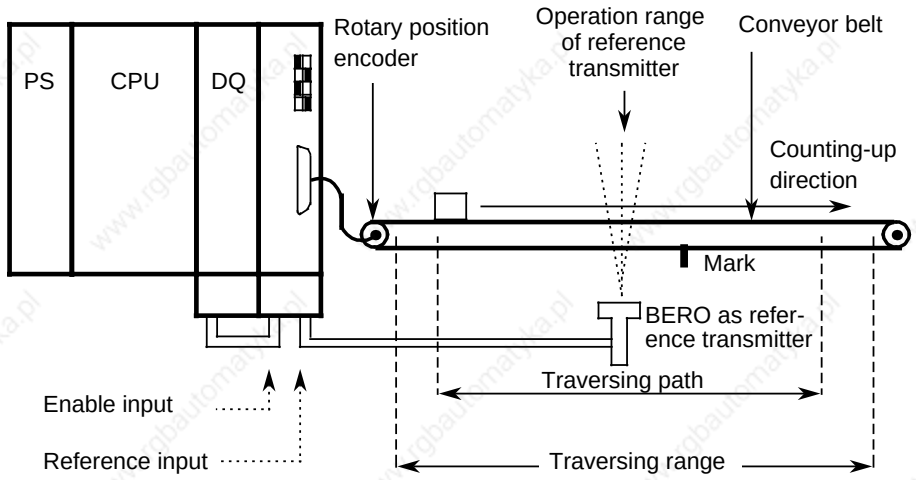



**Figure 15-20. Schematic of a Reference Point Approach Operation**

### Starting the Counter

The counter is reset and started by setting the SYNC bit in the diagnostic byte during the reference point approach operation. The active pulses are counted according to the rotation direction of the position encoder. The count value is incremented during a positive count direction, and decremented during a negative count direction.

### Enabling the Outputs - Reaching the Setpoints - Resetting the Outputs

The two outputs are enabled for switching by a positive pulse edge at the enable input.

An output and the associated diagnostic bit “setpoint reached” are set if all of the following statements are true:

- The position decoder was synchronized (SYNC bit=1 and overflow bit=0).
- The enable signal (terminal 3 on the terminal block) is set to “1” signal.
- The actual value corresponds to the selected setpoint.
  The setpoint can be reached in the up-count or down-count direction.

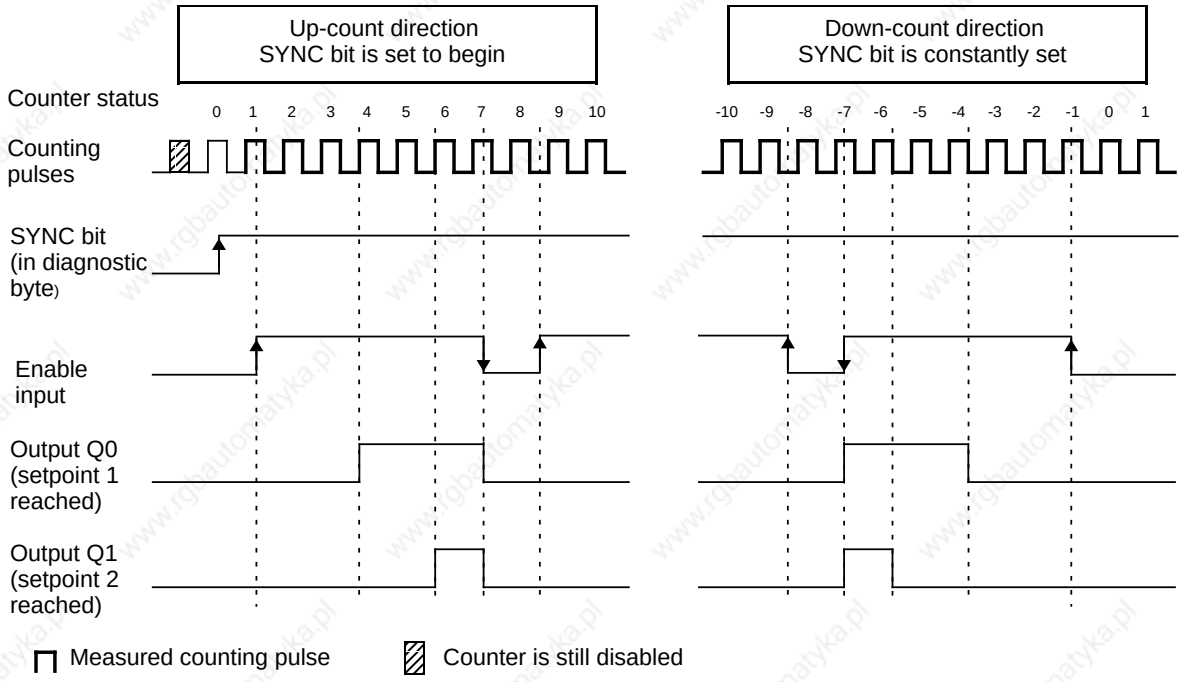



**Figure 15-21. Enabling the Outputs - Reaching the Setpoints - Resetting the Outputs**

After reaching setpoint 1, the output Q 0 is energized and the status bit S1 is set. After reaching setpoint 2, the output Q 1 is energized and status bit S2 is set.

As long as the enable input is active, the outputs are switchable through the module. If the enable command is cancelled, the outputs are switched off and the diagnostic bits are reset. The current actual value is still being measured and incremented or decremented depending on the direction of rotation.

You can read the current count in the STEP 5 program. The actual value is displayed as a signed whole number in two's complement and lies in the range - 32,768 to +32,767.

**Note**

Before you enable the outputs to be switched on by setting the enable input to “1”, make sure the following conditions exist:

- Both setpoints were transferred.
- The overflow bit=0.
- The SYNC bit=1.

If you ignore these prerequisites, the outputs are switched on directly when the actual value=0.

The diagnostic bit and the output are reset with the “0” signal at the enable input. Outputs Q 0 and Q 1 are also reset when the programmable controller goes from RUN to STOP.

The following examples show the switching on of the output at the selected setpoint. There are three possibilities:

- Reaching the setpoint in the direction of a rising actual value
- Reaching the setpoint in the direction of a falling actual value
- Reaching the setpoint in the direction of a rising actual value, then a reversal of direction and a reapproaching of the setpoint in the opposite direction

**Example 1: Approaching a Setpoint in Up-Count Direction**

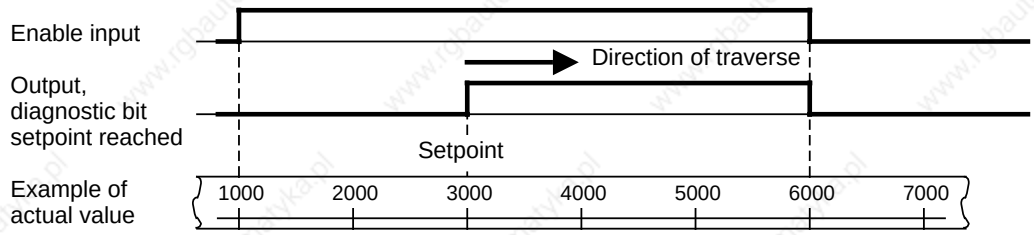



**Figure 15-22. Approaching a Setpoint in Up-Count Direction**

- Actual value=1000: The enable input is set to “1”.
- Actual value=3000: The setpoint is reached, output and diagnostic bit “setpoint reached” are set.
- Actual value=6000: The enable input is set to “0”, output and diagnostic bit are reset.

.

**Example 2: Approaching a Setpoint in Down-Count Direction**

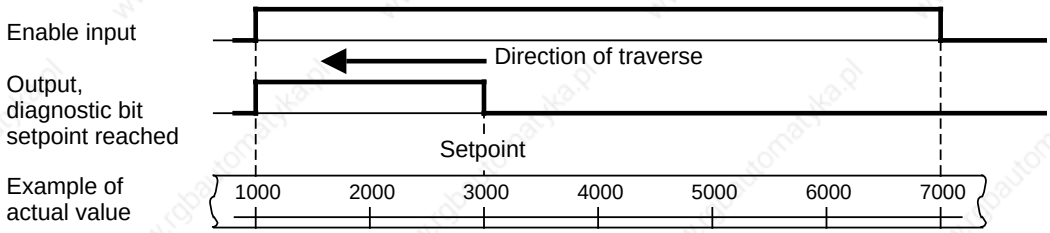



**Figure 15-23. Approaching a Setpoint in Down-Count Direction**

- Actual value=7000: The enable input is set to “1”.
- Actual value=3000: The setpoint is reached, the output and the diagnostic bit “setpoint reached” are set.
- Actual value=1000: The enable input is set to “0”, the output and the diagnostic bit are reset.

**Example 3: Reversal of Direction after Approaching a Setpoint**

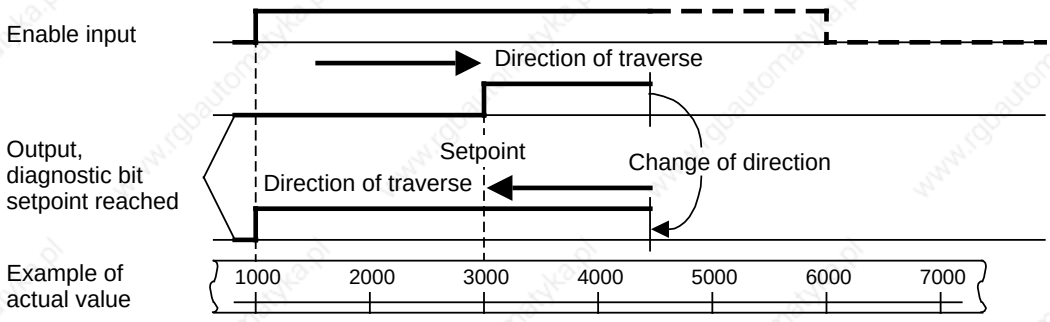



**Figure 15-24. Approaching a Setpoint in Up-Count Direction and Consecutive Reversal of Direction**

- Actual value=1000: The enable input is set to “1”.
- Actual value=3000: The setpoint is reached, the output and the diagnostic bit “setpoint reached” are set.
- Actual value=4500: The traversing path is reversed.
- Actual value=1000: The enable input is set to “0”, the output and the diagnostic bit are reset.

**Note**

Set outputs can be reset only via a “0” signal to the enable input.

**Performance during Overflow**

If the counter leaves the counting range of -32,768 to + 32,767, then the following occurs:

- Bit 3 (overflow) in the diagnostic byte is set to “1”.
- The outputs of the counter module are disabled.

The enable input (terminal 4 of the terminal block) must be set to “0”, in order to switch off active outputs.

After an overflow, a new reference point approach operation has to be executed for synchronization of the actual value detection. After reaching the synchronization, bit 3 in the diagnostic byte is again set to “0”, and the outputs along with the active enable input can be turned on.

**Note**

During an overflow, active outputs are not switched off, and the SYNC bit (bit 0 in the diagnostic byte) is not reset.

### 15.6.5 Entering New Setpoints for the Counter and Position Decoder

Entering new setpoints is always possible via the PIQ. However, a setpoint is only valid if the respective output is not switched on. The status of the outputs is displayed with diagnostic bits S1 and S2.

Diagnostic bit S1 (bit 1 in the diagnostic byte)=1: setpoint 1 is reached and output 1 is switched on.

Diagnostic bit S2 (bit 2 in the diagnostic byte)=1: setpoint 2 is reached and output 2 is switched on.

**Table 15-5. Reaction of the Counter Module during Transfer of the Setpoints**

| Diag. Bit | Response |
|---|---|
| S1 = 0<br>S2 = 0 | New setpoint 1 is transferred and is valid immediately.<br>New setpoint 2 is transferred and is valid immediately. |
| S1 = 1<br><br>S2 = 1 | New setpoint 1 only becomes active if a positive edge has appeared at the enable input.<br>New setpoint 2 only becomes active if a positive edge has appeared at the enable input. |

**Example:**

You want to control a drive by using the outputs of the counter module. After a run of positioning, both setpoints are reached and both outputs are turned on. You can enter the new setpoints by using the following sequence:

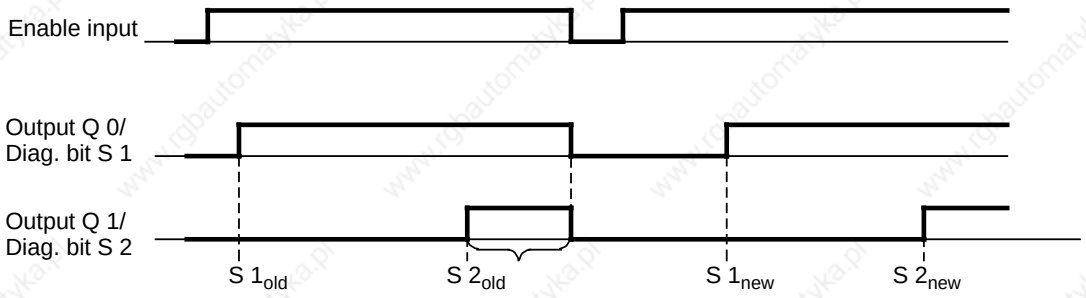



**Figure 15-25. Requirement for New Setpoints**

Transfer the new setpoints to the module. Since both diagnostic bits S1 and S2 are set to “1”, the actual values are **not yet accepted**.

Switch the signal now at the enable input to “0”. With the falling edge, the outputs are switched off and the diagnostic bits are reset.

Switch the signal at the enable input again to “1”. The new **setpoints are accepted and are now active**.

After reaching the new setpoints, the respective output is switched on again.

## 15.6.6 Addressing

The counter module is addressed like an analog module (see section 6.3).

- The module may only be plugged into slots 0 to 7.
- The address range extends from byte 64 to byte 127.
- In both process image tables, eight bytes are reserved per slot and of these eight bytes only the first four are used.

**Slot Addressing**

**Table 15-6. Slot Addressing**

| Slot | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Address PII/PIQ | 64 to 71 | 72 to 79 | 80 to 87 | 88 to 95 | 96 to 103 | 104 to 111 | 112 to 119 | 120 to 127 |

**Meaning of the Bytes of a Slot Address (Example: Slot 1)**

**Table 15-7. Meaning of the Address Bytes of a Slot Address (Example: Slot 1)**

| Byte Number | Byte Address | Meaning in PII | Meaning in PIQ |
|---|---|---|---|
| 0 | 72 | Irrelevant | High byte — Setpoint 1 |
| 1 | 73 | Diagnostic byte | Low byte — Setpoint 1 |
| 2 | 74 | High byte — Actual value | High byte — Setpoint 2 |
| 3 | 75 | Low byte — Actual value | Low byte — Setpoint 2 |
| 4 to 7 | 76 to 79 | Irrelevant | |

**Examples for Data Exchange between the Programmable Controller and the Counter Module**

**Example 1:**

The counter module is plugged into slot 4. If you now wish to check whether your system for position decoding has been synchronized by a reference point approach, you must scan the sync bit in the diagnostic byte (bit 0). If this bit is set, a branch is to be made to FB20. The position decoding operation is started in FB20.

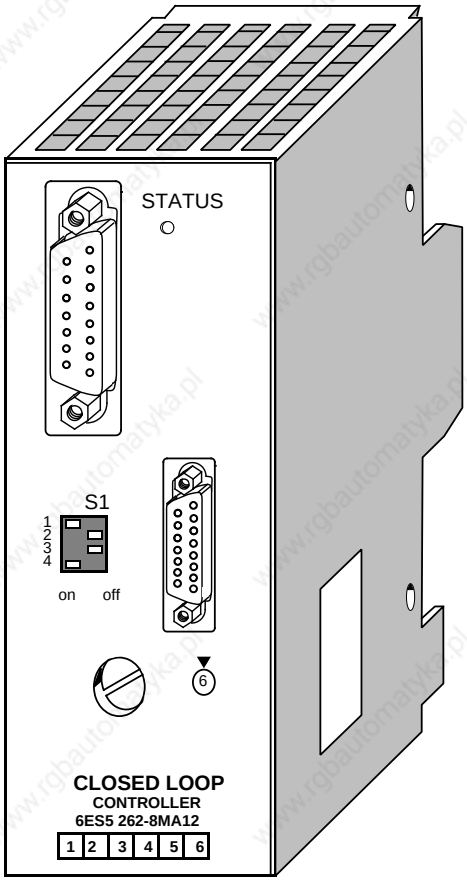
| STL | Description |
|---|---|
| ...<br>A I 97.0<br>JC FB 20<br><br>... | Read in bit 0 of the diagnostic byte (sync bit).<br>If this bit is set, a branch is made to FB20.<br>If the bit is not set, program scanning is continued with the statement following the block call. |

**Example 2:**

Transferring the setpoints stored in flag words 0 and 2 to the counter module inserted into slot 7. The module has only to accept the setpoints when the old setpoints have been reached or exceeded.

| STL | | | | Description |
|---|---|---|---|---|
| ... | | | | |
| | AN | I | 121.1 | If setpoint 1 has not yet been reached (bit 1=0), a branch is made to label 1. |
| | JC= | L001 | | |
| | L | FW | 0 | Read in setpoint 1 |
| | T | QW | 120 | and transfer it to the counter module. |
| L001 | AN | I | 121.2 | If setpoint 2 has not yet been reached (bit 2=0), a branch is made to label 2. |
| | JC= | L002 | | |
| | L | FW | 2 | Read in setpoint 2 |
| | T | QW | 122 | and transfer it to the counter module. |
| L002 | BE | | | Block end |
| ... | | | | |

## 15.7 Closed-Loop Control Module IP 262 (6ES5 262-8MA12) (6ES5 262-8MB12)



**Technical Specifications**

**Controller**

| | |
|---|---|
| Total cycle time (equals scan time) | 100 to 200 ms |
| Resolution of the open-loop controller | 5 ms at 50 Hz<br>4.2 ms at 60 Hz |

**Analog Inputs**

| | |
|---|---|
| Number of inputs | 4 (suited for current, thermo-couple, or resist-ance thermometer), voltage with external switching |
| Additional input for reference temperature | 1 (resistance thermometer) |
| Galvanic isolation | no |
| Permissible voltage difference | |
| - between inputs | - 1 V to +1 V |
| - between inputs and central ground point | - 1 V to +1 V |
| Digital representation of the input signal | 11 bits+sign |
| Current input | |
| - input signal range | 0 to 20 mA or<br>4 to 20 mA |
| - input resistance | 24.3 ±0.1% |
| mV Input (for thermocouple) | |
| - input signal range | 0 to 50 mV or<br>- 8.9 to 41.1 mV<br>(type J, K, L, S) |
| Cable impedance | 30 per wire |
| Resistance thermometer | |
| - start | 18.49 |
| - end | 219.12 |
| - permissible cable impedance | 30 per wire |

**Binary Inputs**

| | |
|---|---|
| Number of inputs | 4 |
| Galvanic isolation | no |
| Signals state “0” | - 30 to+4.5 V or open |
| Signal state “1” | +13 to +30 V (signal state invertible) |
| Input resistance | approx. 4 k |

**Analog outputs of the constant controller** (6ES5 262-8MA12)

| | |
|---|---|
| Number of outputs | 3 |
| Galvanic isolation | no |
| Output signal range | 0 to 20 mA or<br>4 to 20 mA |
| Maximum permissible load | 600 |
| No load voltage | (L+) - 2 V |

**Binary outputs for the open-loop controller** (6ES5 262-8MB11)

| | |
|---|---|
| Number of outputs | 8 |
| Galvanic isolation | no |
| Signal state “0” | <1.5 V |
| Signal state “1” | (L+) - 3.8 V |
| Maximum load current | 100 mA short-circuit proof |

**Wiring method**

| | |
|---|---|
| Programmer (PG)<br>Operator panel (OP)<br>SINEC L1 network connection | front side via 15-pin subminiature D connector |
| Connectable are | PG 605, PG 635, PG 675, PG 685, PG 695, PG 730, PG 750, OP 393, OP 396, OP 395 |
| Analog and binary inputs | front side via 25-pin subminiature D connector |
| Analog and binary outputs | via terminal block of the bus unit |

**General data**

| | |
|---|---|
| Input voltage | |
| - rated value | 24 V DC |
| - permissible range | 18 to 34 V DC |
| - permissible range with the PG 605/OP 393 | 18 to 27 V DC |
| Current consumption | |
| - internal (from the CPU; 9 V) | approx. 20 mA |
| - external (for 24 V; without load) | approx. 180 mA |
| - external (for 24 V; without load; with PG 605/OP 393) | approx. 340 mA |
| Ambient temperature | 0 ° to 55 °C (32 to 131°F) |

**Function**

The S5-100U programmable controller offers different solutions for individual closed-loop control tasks. First there is a software solution for CPU 103, version 8MA02 and higher, via function blocks. Second, there is a control module solution (for example, a module that can solve PID control tasks simply and in a time saving manner). The basis, in both cases, is a PID control algorithm.

The closed-loop control module IP 262 can be used with the S5-90U, S5-95U, and S5-100U programmable controllers. It can be used without COM software.

The module relieves the programmable controller from closed-loop control tasks. The IP 262 also works with its own power supply in a stand-alone operation. The module can function independently without a programmable controller and can handle up to four closed-loop control circuits.

Two interfaces are located on the front panel of the module.

- An interface for the connection of a programmer (PG) or an operator panel (OP) or the SINEC L1 Network (under development)
- An interface for the connection of analog and binary inputs

In addition, the following items are available:

- A selector switch for each channel for current and voltage (thermocouples or PT 100)
- A status LED for RUN (a continuously lit green light), transducer malfunction (blinking light), and module malfunction (off)

The module is well suited to take over control-loop tasks in the area of industrial processing technology, for example, temperature control, pressure and flow control, continuous injection functions, and non-time-critical closed-loop rpm controls.

**Modules**

There are two IP 262 modules.

- ... - 8MA12 with 3 analog outputs for continuous controllers with analog output signals
- ... - 8MB12 with 8 binary outputs for continuous controllers with pulse time-interval signals or for step-action controllers

Additionally, the module provides the following inputs:

- 4 analog inputs for direct feed of setpoint and actual values
- 4 binary inputs for control variables

**Installation**

- The closed-loop control module is plugged into a bus unit like any other input or output module (see chapter 3).
- The module can only be plugged into slots 0 to 7.
- The connections for power supply and the analog and binary output signals are located on the terminal block of the bus unit.
- The analog and binary inputs are connected to the module with a 25-pin sub-D female connector.

### Addressing

The module is addressed like a four-channel analog module.

### Operating Modes

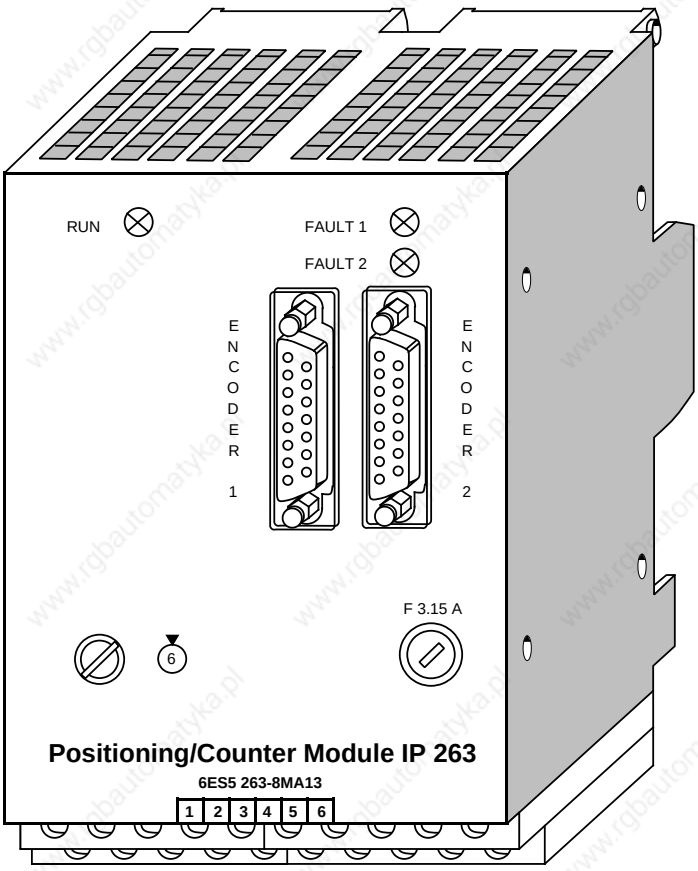
Since transducers and sensors are directly wired to the module, the module can work independently from a programmable controller in stand-alone operation, provided that the setpoints and the 24-V power supply voltage are fed directly to the IP 262. This means that the module executes the control and the output of the manipulated variable and can work alone or be controlled via the SINEC L1 by a master unit.

Besides this, the IP 262 has its own back-up, which means that the module can continue to work alone in the event the master CPU (e.g., S5-135U with R64) fails. It uses the last setpoint received from the CPU or the predefined back-up setpoint.

Two operating modes are possible:

- DDC Operation (Direct Digital Control)
  The control is executed entirely from the CPU. The IP only outputs the manipulated variable. If the CPU fails, the module can continue to control independently with a predefined back-up setpoint.
- SPC Operation (SetPoint Control)
  The module receives only the setpoint from the CPU; the control task is carried out independently of the CPU. If the CPU fails, the IP continues to control using the last setpoint received from the CPU. It is also possible to use a predefined back-up setpoint here.

## 15.8 IP 263 Positioning Module (6ES5 263-8MA13)



**Technical Specifications**

**Encoders**

| | |
|---|---|
| Position decoder | incremental, absolute (SSI interface) |
| Maximum traversing range | |
| - with incremental encoders | $2^{24}$ increments |
| - with absolute encoders | 8192 increments per revolution × 2048 revolutions |
| Signal voltages | |
| - Differential inputs | 5 V to RS 422 |
| - Asymmetrical inputs | 24 V (only incremental encoders) |
| Supply votlage for encoders (short-circuit-proof, no overload) | 5 V/300 mA<br>24 V/300 mA |
| Input frequency and cable length | |
| Symmetrical encoders (5 V signals): | |
| - with 5 V encoder supply | max. 200 kHz for 32 m (105 ft.) cable, shielded |
| - with 24 V encoder supply | max. 200 kHz for 100 m (328 ft.) cable, shielded |
| Asymmetrical encoders (24 V signale): | max. 100 kHz for 25 m (82 ft.) cable, shielded<br>max. 25 kHz for 100 m (328 ft.) cable, shielded |
| Data transmission rate and cable length with absolute encoders | 62.5 kHz (selectable in steps)<br>125 kHz (160 m/ 525 ft. shielded)<br>250 kHz<br>500 kHz<br>1 MHz (32 m/ 105 ft. shielded) |
| Input signals | |
| - Incremental | 2 pulse trains displaced by 90°<br>1 zero pulse |
| - 24 V initiator (BERO) | 1 pulse train |
| - SSI | Absolute value |
| Input currents | to RS 422 |
| - 5 V | typ. 5 mA |
| - 24 V | |

**Digital Inputs**

| | |
|---|---|
| Input voltage range | - 3 V to + 30 V |
| Galvanic isolation | no |
| 0 signal | - 3 V to +5 V |
| 1 signal | +13 V to+30 V |
| Permissible zero-signal current at 0 signal | 1.1 mA |
| Input current at 24 V | typ. 5 mA |

Other: If the digital inputs are used, they must always be connected to a defined potential (0 V, 24 V) and must not be kept open.

**Digital Outputs**

| | |
|---|---|
| Output voltage range | +20 V to+30 V |
| Galvanic isolation | no |
| Output current at 1 signal | max. 500 mA |
| Short-circuit protection | Short-circuit-proof output |
| Cable length, shielded | max. 100 m (328 ft.) |

**Supply Voltage**

| | |
|---|---|
| Logic voltage from 24 V supply produced with switched-mode power supply | 4.9 V to 5.1 V |
| Current consumption from 24 V without outputs and encoder | typ. 120 mA |
| Undervoltage monitoring | $V_{internal} < 4.65$ V |

**Power Loss** typ. 4 W

A separate manual is available for the IP 263 positioning module. It can be ordered under the order number 6ES5 998-5SK21.

The IP 263 is suitable for positioning of two independent axes.

**Assignments of Outputs**

The IP 263 is a two-channel module: 4 digital outputs are assigned to each channel for the control of drives;

- Rapid traverse
- Creep speed
- Anti-clockwise rotation
- Clockwise rotation

Both incremental and absolute encoders (SSI - synchronous serial interface) can be connected for actual position encoding.

They transmit the machine data, such as

- Software limit switches
- Resolution
- Cutoff difference
- Switchover difference
- Zero-speed control

The syntax for the data block which has to be created for this purpose is simple and is described in the manual.

**Positioning**

The only thing that remains to be done is to specify the desired target and then the module is ready for the positioning procedure.

The IP 263 then carries out positioning automatically. When the target has been reached, it sends a message to the IM 318-B interface module and thus to the CPU.

Figure 15-26 shows the positioning procedure with the IP 263: After the start, a rapid traverse towards the target takes place first. When the switchover/cutoff point has been reached, a switch-over to creep speed or cutoff takes place. Afterwards, the IP 263 monitors approach of the target. When the axis has reached the target range, a signal is sent to the IM 318-B interface module.

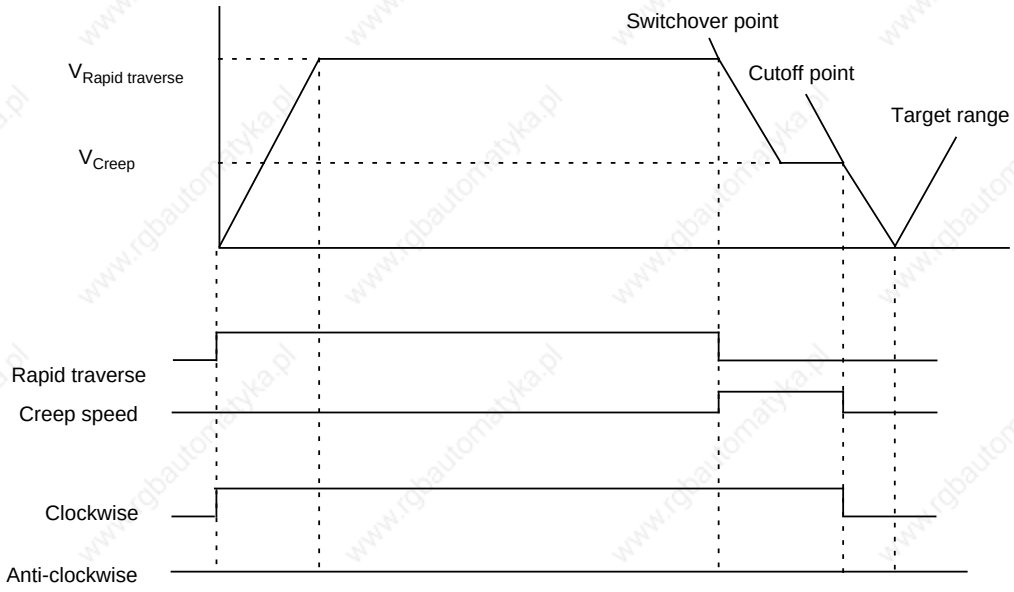



**Fig. 15-26. Positioning with the IP 263**

During reference point travel, the digital input of the module senses the speed reducing cam (reference point switch).

In the ”Length measurement” operating mode, the module senses encoder pulses as long as this input has a ”1” signal.

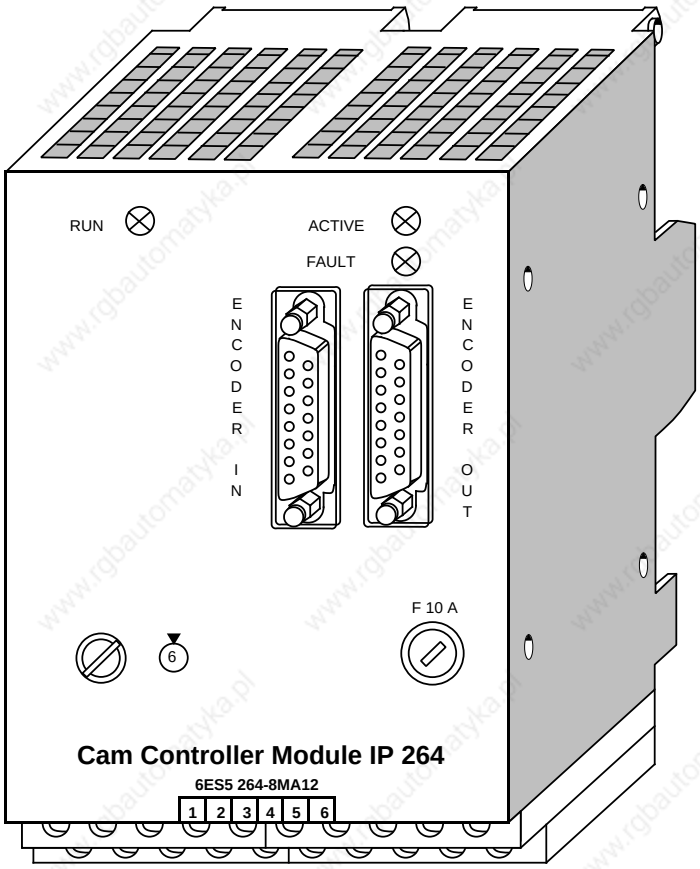
**Installation**

As other I/O modules, the IP 263 is mounted on the bus unit (see chapter 3).

**Addressing**

The IP 263 is addressed like a 4-channel analog module.

## 15.9 IP 264 Electronic Cam Controller Module 6ES5 264-8MA12



**Technical Specifications**

**Encoders**

| | |
|---|---|
| Actual value sensing | incremental, absolute (SSI interface) |
| Maximum traversing range | |
| - with incremental encoders | $2^{16}$ increments |
| - with absolute encoders | $2^{16}$ encoders |
| Signal voltages | |
| - Differential inputs | 5 V to RS 422 |
| - Asymmetrical inputs | 24 V (only incremental encoders) |
| Supply voltage for encoders (short-circuit-proof, no overload) | 5 V/300 mA<br>24 V/300 mA |
| Input frequency and cable length | |
| Symmetrical encoders (5 V signals): | |
| - with 5 V encoder supply | max. 200 kHz for 32 m (105 ft.) cable, shielded |
| - with 24 V encoder supply | max. 200 kHz for 100 m (328 ft.) cable, shielded |
| Asymmetrical encoders (24 V signal): | max. 100 kHz for 25 m (82 ft.) cable, shielded<br>max. 25 kHz for 100 m (328 ft.) cable, shielded |
| Data transmission rate and cable length with absolute encoders (selectable in steps) | 125 kHz (160 m/ 525 ft. shielded)<br>250 kHz<br>500 kHz<br>1 MHz (32 m/ 105 ft. shileded) |
| Input signals | |
| - Incremental | 2 pulse trains displaced by 90°<br>1 zero pulse |
| - 24 V initiator (BERO) | 1 pulse train |
| - SSI | Absolute value |
| Input currents | to RS 422 |
| - 5 V | typ. 5 mA |
| - 24 V | |

**Digital Inputs**

| | |
|---|---|
| Input voltage range | -3 V to + 30 V |
| Galvanic isolation | no |
| 0 signal | - 3 V to +5 V |
| 1 signal | +13 V to+30 V |
| Permissible zero-signal current at 0 signal | 1.1 mA |
| Input current at 24 V | typ. 5 mA |

Other: If the digital inputs are used, they must always be connected to a defined potential (0 V, 24 V) and must not be kept open.

**Digital Outputs**

| | |
|---|---|
| Output voltage range | +20 V to+30 V |
| Galvanic isolation | no |
| Output current at 1 signal | max. 300 mA |
| Short-circuit protection | Short-circuit-proof output |
| Cable length, shielded | max. 100 m (328 ft.) |

**Supply Voltage**

| | |
|---|---|
| Logic voltage from 24 V supply produced with switched-mode power supply | 4.9 V to 5.1 V |
| Current consumption from 24 V without outputs and sensors | typ. 120 mA |
| Undervoltage monitoring | $V_{internal} < 4.65$ V |

**Power Loss** typ. 4 W

**Module Cycle Time (incl. dead-time compensation)**

| | |
|---|---|
| Separate cam programs with max. 32 cams each for forwards and backwards (incl. dead-time compensation) | 57.6 µs |
| "Common" cam program with max. 32/64 cams for forwards and backwards | 57.6/115.2 µs |

A separate manual is available for the electronic cam controller. It can be ordered under the order number 6ES5 998-5SL21.

The IP 264 can be used both for rotary and linear axes.

The IP 264 electronic cam controller makes electronic processing of cams economical even for applications in the lower performance range.

32 cams which can be allocated as desired to 16 tracks have a switching accuracy of better than 1 degree at 2400 revolutions per minute. This corresponds to a response time of less than 60 µs. For applications with low precision requirements it is even possible to program 64 cams.

It is also possible to integrate 32 cams each into a cam program for ”forwards” and a cam program for ”backwards”. Switchover between these two programs is carried out by automatic direction sensing of the IP 264 or it is controlled by the SIMATIC S5.

All cams can be defined either as path-path cams or as path-time cams.

**Dead Time Compensation**

Through the speed-dependent, dynamic shift, each individual cam compensates the dead time of the actuator connected (e.g. pneumatic valve) at a scanning rate of 60 µs. This enables the utmost accuracies to be achieved even at changing drive speeds.

**Direct Process Connection**

In order to be able to pass on the short response time of the IP 264 directly to the process, a digital output (24 V, 0.3 A) is available on the module for each track. Generally, the units to be controlled can be connected directly. Auxiliary contactors are required only for actuators with a higher current consumption.

The sensors to be connected can be incremental encoders, absolute SSI encoders (SSI= synchronous serial interface) or simple 24 V signal sensors (e.g. BEROs). The sensor data can be looped through to further modules via the additional sensor output, without separating the sensor cables mechanically or using additional fan-out units.

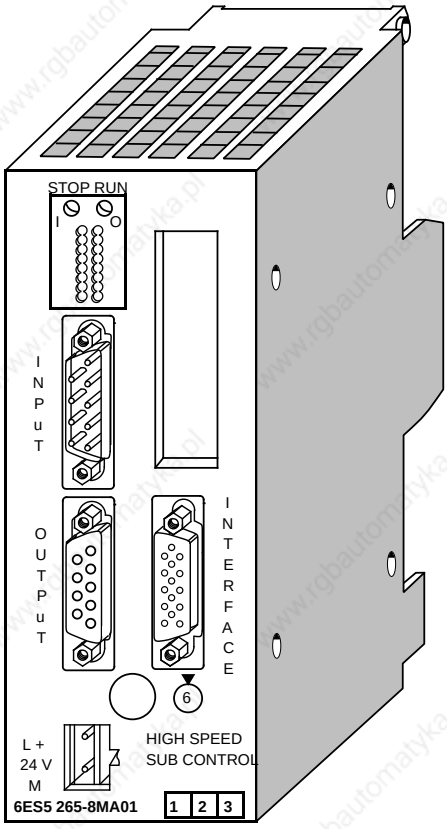
**Installation**

As other I/O modules, the IP 263 is mounted on the bus unit (see chapter 3).

**Addressing**

The IP 264 is addressed like a 4-channel analog module.

## 15.10 IP 265 High Speed Sub Control (6ES5 265-8MA01)



**Technical Specifications**

| | |
|---|---|
| Current consumption from +9 V (CPU) | <175 mA |
| Signal status display | only for 24 V inputs and 24 V outputs (green LEDs) |
| Operating status display | STOP (red LED)<br>RUN (green LED) |
| Memory submodule | EPROM/EEPROM |
| Power loss | typ. 2.3 W |
| Weight | approx. 300 g (10.6 oz.) |

**Digital 24 V Inputs (9-pin sub D connector)**

| | |
|---|---|
| Number of inputs | 8 |
| Galvanic isolation | no |
| Status display | Yes, on 5 V side |
| Input votlage L+ | |
| - Rated value | 24 V DC |
| - for "0" signal | 0 to 5 V |
| - for "1" signal | 11 to 30 V (IEC 65A) |
| Input current at "1" signal | typ. 6.5 mA (IEC 65 A) |
| Connection of 2-wire BERO | possible (zero signal current 1.5 mA) |
| Input frequency | max. 10 kHz |
| Cable length (shielded) | max. 100 m (330 ft.) |
| Delay time of input circuit | |
| - Rising edge | typ. 15 µs |
| - Falling edge | typ. 10 µs |

**5 V differential inputs (15-pin D sub HD socket connector)**

| | |
|---|---|
| Number and type of input signals | 3 differential signals RS 422 |
| Input frequency | max. 58 kHz |
| Pulse length | |
| - "Low" level | min. 8.6 µs |
| - "High" level | min. 8.6 µs |
| Cable length (shielded) | max. 32 m (105 ft.) |

**Digital 24 V outputs (9-pin sub D socket connector)**

| | |
|---|---|
| Number of outputs | 8 |
| Galvanic isolation | no |
| Status display | Yes, on 5 V side |
| Short-circuit protection | Yes, electronic clock cycle |
| Load voltage L+ | |
| - Rated value | 24 V DC |
| - Permissible range | 20 to 30 V |
| Output current at "1" signal | 0.5 A at 60 °C |
| Permissible total current of output | 2 A at 60 °C |
| Connection of outputs in parallel | possible in pairs ($I_{outp}$=0.8×$I_{rated}$) |
| Output frequency at ohmic load | max. 1 kHz at 15 mA load*<br>max. 2 kHz at 50 mA load*<br>max. 4 kHz at 500 mA load* |
| Cable length | max. 100 m (330 ft.) |
| Lamp load | max. 2 W |
| Residual current at "0" signal | max. 1 mA |
| Voltage drop at "1" signal | max. 1 V |
| Limitation of inductive cut-off voltage | -15 V |
| Delay time of output circuit | |
| - Rising edge | typ. 10 µs |
| - Falling edge, depending on ohmic load: | typ. 150 µs at 15 mA load*<br>typ. 90 µs at 50 mA load*<br>typ. 70 µs at 500 mA load* |

**Expansion input and outputs (15-pin D sub HD socket connector)**

| | |
|---|---|
| Number of inputs and outputs | 8 (any desired mixture of I/Os can be configured) |

**Connector for 24 V load voltage (2-pin)**

| | |
|---|---|
| Permissible cross-sections of cables | |
| - Flexible cable H07V-K with end sleeve | 0.5 to 1.5 mm$^2$ |
| - Solid cable H07V-U | 0.5 to 2.5 mm$^2$ |

***** Peak value (no effective value specified)

The IP 265 High Speed Sub Control is a powerful, user-programmable I/O module which relieves the CPU's of the SIMATIC S5-100 systems of automation tasks which place great demand on speed and reproducibility.

A separate manual is available for the IP 265. It can be ordered under the order number 6ES5 998-5SH21.

**Function**

The IP 265 High Speed Sub Control is available with a COM software package which is required to determine the function of the module.

The use of the IP 265 in an S5 system enables rapid I/O processing in the millisecond range. By implementation of an FPGA (Field Programmable Gate Array) in the IP 265 it is possible to process process signals in parallel and very fast.

The IP 265 user program consists of elementary basic functions such as logic operations, counters, timers or comparators. The structure of the IP 265 user program is based on the CSF5 type of representation.

The following can be used:

- Either a user-programmed user program
  or
- a fixed-programmed standard program from SIEMENS.

The COM 265 is available for user-programming of the IP 265. Besides it being programmable, the IP 265 can also be used to implement the special ”counter” function with a fixed-program standard program. For this purpose, SIEMENS AG offers a memory submodule for the IP 265 with the standard ”counter” function.

The IP 265 user program is automatically processed by the IP 265. It conditions process input signals to process output signals. The IP 265 can read 11 process inputs (8 x 24 V inputs, 3 x 5 differential inputs) and set 8 process outputs (24 V outputs).

The program capacity of the FPGA and the number of process inputs/outputs of **one** IP 265 are limited. The IP 265 is therefore used for rapid sub controls. By adding one IP 265 to another, complex sub-processes can be controlled with this module.

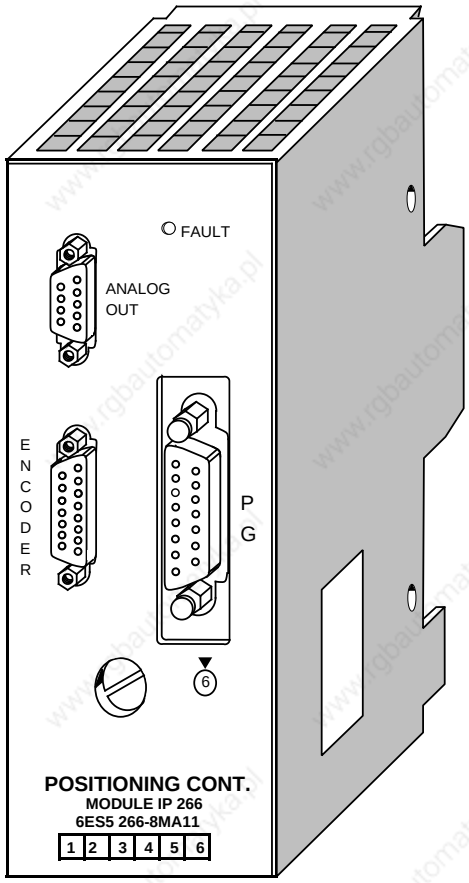
**Installation**

As other I/O modules, the IP 265 High Speed Sub Control is mounted on a bus unit.

**Adressing**

The module is addressed like a 4-channel analog module.

## 15.11 Positioning Module IP 266 (6ES5 266-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| **Analog Output** | | |
| Output signal range | | ±10 V |
| Digital signal representation | | 13 bits plus sign |
| Short-circuit proof | | yes |
| Reference potential of the analog output signal | | analog ground of the power section |
| Cable length shielded | max. | 32 m (105 ft.) |
| **Pulse Input** | | |
| Position decoder | | incremental |
| Traverse range | | ±32767.999 mm/ 0.1 inch/degree |
| Input voltages for the tracks<br>- differential inputs<br>- asymmetrical inputs | | <br>5 V/RS 422<br>24 V/typ. 7.3 mA |
| Supply voltage for the sensor (short-circuit proof) | | 5 V/350 mA<br>24 V/350 mA |
| **Input Frequency and Cable Length** | | |
| Symmetrical sensors (5 V) | max.<br>max. | 500 kHz,<br>30 m (98 ft.) shielded cable length |
| Asymmetrical sensors (24 V) | max. | 100 kHz for 25 m (82 ft.) cable length shielded |
| | max. | 25 kHz for 100 m (330 ft.) cable length shielded |
| **Input Signals** | | 2 pulse series 90 degrees out of phase 1 zero pulse |
| **Digital Inputs** | | |
| Input voltage range | | ±30 V |
| Galvanic isolation | | no |
| “0” signal | | - 30 V to +5 V |
| “1” signal | | 13 V to 30 V |
| Permissible zero signal current at “0” signal | | 1.5 mA |
| Typ. input current at 24 V | | 7.3 mA |
| **Digital Outputs** | | |
| Output voltage range | | 20 V to 30 V |
| Galvanic isolation | | no |
| Max. output current at “1” signal | | 100 mA |
| Short-circuit protector | | short-circuit proof output |
| Cable length shielded | max. | 100 m (330 ft.) |
| **Supply Voltage** | | |
| Logic voltage from 24-V ext. supply produced with switched-mode power supply | | 4.7 V to 5.5 V |
| Current consumption from 24-V supply without outputs and 24-V sensor | typ. | 180 mA |

Because of its performance capability and the complexity of its description, the IP 266 has its own manual that you can order separately. The order number is: 6ES5 998-5SC21. The positioning control module IP 266 expands the field of application for “positioning operations” of the S5-100U.

As an “intelligent I/O module”, it allows you to use open-loop as well as closed-loop control positioning.

The positioning operations are processed independently of the execution times of the user programs in the programmable controller. Thus the CPU is not burdened with positioning jobs constantly being processed. You can plug the IP 266 into slots 0 to 7 on the S5-100U. The IP 266 is assigned addresses in the analog address area of the programmable controller.

**Operation Principle of the IP 266**

The IP 266 enables you to control the positioning operation of your drive exactly.

The module delivers a voltage setpoint in the range of±10 V via an analog output for the control of a power section for servo motors.

The IP 266 needs exact data about your drive system in order to calculate speed, acceleration, or traverse residual distances. This data can be stored in an EEPROM that is permanently installed in the programmable controller. By using its own start-up routine, this data can be accessed immediately after you switch on the programmable controller and can be processed directly.

The IP 266 allows you to select between a linear axis and a circular axis. You can also select the unit of measurement for processing the data: either [mm], [in.] or [deg].

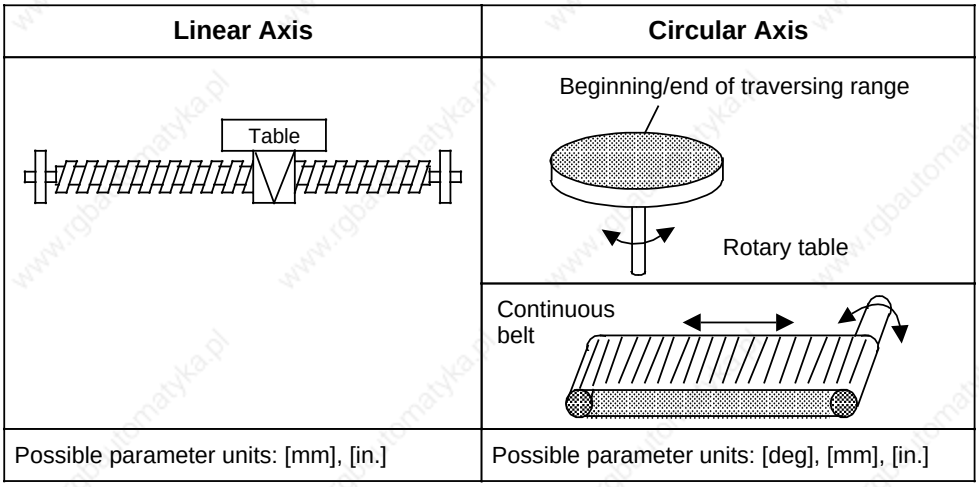



**Figure 15-27. Units of Measurement that IP 266 Can Process for Circular Axis and Linear Axis**

Besides purely traversing movements, other operating modes allow offset generation of axis coordinates or drift compensation in the system.

In addition, the IP 266 offers operating modes to read data such as positioning actual value or residual traversing distances.

In order to use the IP 266 in an automatic manufacturing process, it is possible to combine individual traversing applications, positioning corrections, offsets or dwell times in a “traversing program”. These traversing programs can be called up via two special operating modes and processed automatically or semi-automatically.

Such a traversing program can be created by using the “learning capable” “Teach-in mode” for positioning applications. The information from single positioning applications can be stored at the end of an operation in a traversing program.

**Positioning**

For the positioning operation, the IP 266 calculates the setpoint from the selected end data and velocity data in conjunction with the programmed machine data. The actual value follows the selection. The deviation (following error) that occurs reaches a constant value after the short start-up phase and must reach zero at the end of the positioning operation.

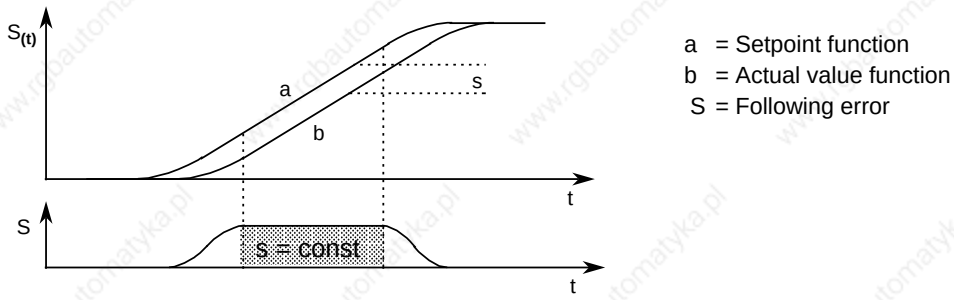



**Figure 15-28. Course of a Following Error during a Positioning Operation**

### Overview of the Operation Modes

**Table 15-8. Designation of the Operating Modes**

| List of the Operating Modes | | |
|---|---|---|
| JOG 1 | AUTOMATIC SINGLE BLOCK | ACKNOWLEDGE ERROR |
| JOG 2 | TEACH-IN ON | DRIFT COMPENSATION ON |
| CONTROLLED JOG | TEACH-IN OFF | DRIFT COMPENSATION OFF |
| FOLLOW-UP MODE | ZERO OFFSET ABSOLUTE | RAM EEPROM |
| REFERENCE POINT | ZERO OFFSET RELATIVE | READ ACTUAL POSITION |
| INCREMENTAL ABSOLUTE | CLEAR ZERO OFFSET | READ FOLLOWING ERROR |
| INCREMENTAL RELATIVE | TOOL OFFSET ON | READ DISTANCE TO GO |
| AUTOMATIC | TOOL OFFSET OFF | SYNCHRONIZE IP |

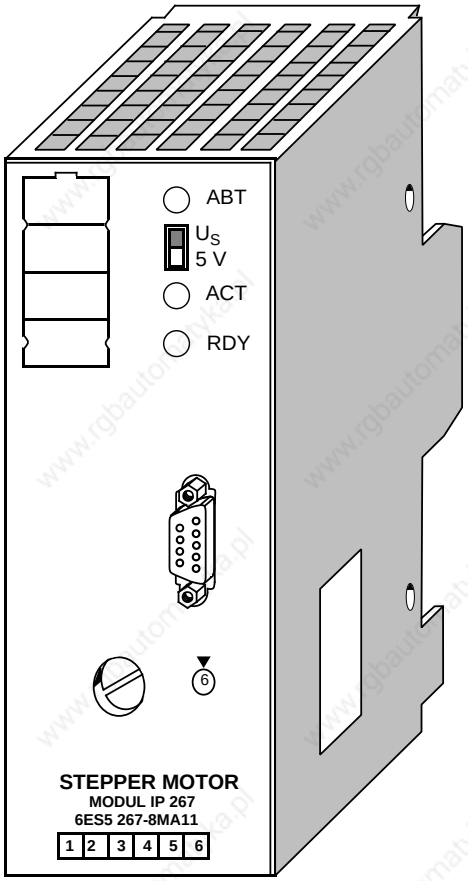
The COM 266 software package offers user friendly operation and programming. The IP 266 exchanges all data with the programmable controller via a serial interface. All tasks written in 8-byte messages are sent to the IP 266 during the program cycle via the process output image table (PIQ). The IP 266 transmits feedback messages cyclically via the process image input table (PII). These messages can be about the actual value position, remaining traversing distance, or following error as well as a status byte, error byte, the current operation mode, and special data from the traversing program.

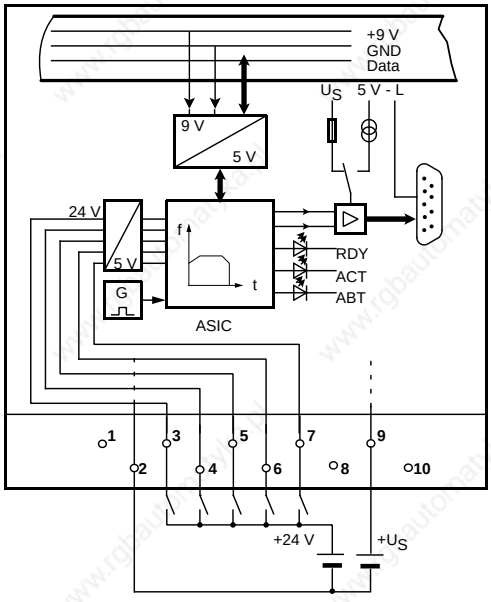
### Installation

1. Plug the IP 266 into a bus unit like any other I/O module (see chapter 3).
2. Insert the IP 266 only into slots 0 to 7.
3. Connect the external switches to the digital inputs of the IP 266 via the terminal block. These switches are used to limit the traversing range. They also allow you to intervene at any time into the processing of the module.
   - The IP 266 can bypass the STEP 5 OB1 cycle, via three digital outputs, and send signals directly to external I/Os. The controller must, however, be enabled (function signal enable controller, FUM) and must be connected to the power section of the drive.
4. Connect the servo motor's power section to the 9-pin subminiature D female connector.
5. Connect the incremental encoder to the left 15-pin subminiature D female connector ENCODER.

You can connect a programmer with screen to the 15-pin subminiature D female connector on the right side to operate the IP 266 via the COM software.

## 15.12 Stepper Motor Control Module IP 267 (6ES5 267-8MA11)




**Technical Specifications**

| | | |
|---|---|---|
| Supply voltage (BUS) | | 9 V |
| Current consumption | approx. | 150 mA |
| Special voltage $V_S$ | | 5 V to 30 V |
| **Digital Inputs** | | |
| Rated input voltage | | 24 V |
| Galvanic isolation | | no |
| Input voltage:<br>“0” signal<br>“1” signal | | <br>- 33 V to 5 V<br>13 V to 33 V |
| Input current | typ. | 8.5 mA |
| Supply voltage for two-wire BEROs | | 22 V to 30 V |
| **9-pin Subminiature D Connector** | | |
| Output voltage with 5-V supply<br>“0” signal<br>“1” signal | <br>max.<br>min. | <br>0.4 V<br>4.5 V |
| With special supply voltage $V_S$ (5 V to 30 V)<br>“0” signal<br>“1” signal | <br>max.<br>min. | <br>0.4 V<br>$V_S$ - 0.4 V |
| Output current | | 20 mA (short-circuit proof) |
| Output frequency | max. | 204 kHz |
| Increment number of steps | max. | $2^{20}$ - 1 pulses/ job |
| Permissible cable length | max. | 50 m (165 ft.) at 50 kHz (twisted pair cable) |

Because of its performance capability and the complexity of its description, the IP 267 has its own manual that you can order separately. The order number is: 6ES5 998-5SD21. The IP 267 Stepper Motor Control Module expands the field of application as an intelligent I/O module (IP) of the S5-100U and S5-95U programmable controllers for “closed-loop control positioning". The IP 267 controls positioning processes independently of the run time of user programs in the programmable controller. The CPU is not loaded with processing positioning job operations.

You can plug the IP 267 into slots 0 to 7 in the programmable controller. It then occupies addresses in the analog address area of the programmable controller.

**Principle of Operation**

The IP 267 generates pulses for the stepper motor power section. The number of output pulses determines the length of the traversing path. The pulse frequency is a measure of the velocity. Each pulse causes the stepper motor shaft to turn through a certain angle. In the case of high-speed pulse trains, this step movement becomes a constant rotational movement. Stepper motors can reproduce all movement sequences only as long as no steps are lost. Step losses can be caused when load variations occur or when the programmed pulse trains exceed motor-specific values.

To enable the IP 267 to generate these pulse trains, the user must enter the following data:

- Configuration data: This data describes the individual stepper motors and the technical characteristics of the drive system.
- Positioning data: This data describes the individual traverse jobs and indicates the velocities, directions, and lengths of the configured paths.

The IP exchanges data with the programmable controller via the serial interface. During the program scans, all necessary information is sent from the process image output table (PIQ) to the IP 267 in 4-byte messages. The IP 267 cyclically transmits feedback signals on the remaining distance to go and various status bits to the process image input table (PII).

Using the configuration and positioning data settings, the IP 267 generates a symmetrical traverse profile consisting of an acceleration ramp, a constant velocity range, and a deceleration ramp.

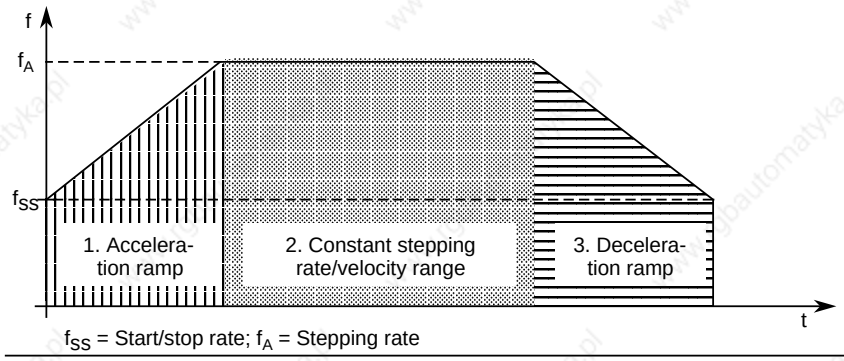



**Figure 15-29. Velocity Profile of the IP 267**

Using a limit switch on the digital inputs, IP 267 can monitor the limits of a traversing range and stop the traversing movement when the permissible range limit is exceeded.

The activated input “external stop” causes a calculated decelerating of the traversing movement.

An emergency limit switch can be installed at input “IS” (pulse inhibit). When this switch responds, the pulse output is interrupted immediately.

For a reference point approach operation, an additional switch can be connected at input REF that lies within the traversing zone. The reference point approach operation is also possible without this switch.

Status LEDs provide you with the following information:

- The IP 267 is configured RDY
- Pulse outputs during a positioning operation ACT
- Interruption of the positioning operation ABT

There are four operating modes:

- STOP
- START FORWARDS
- START BACKWARDS
- NEUTRAL

**Installation**

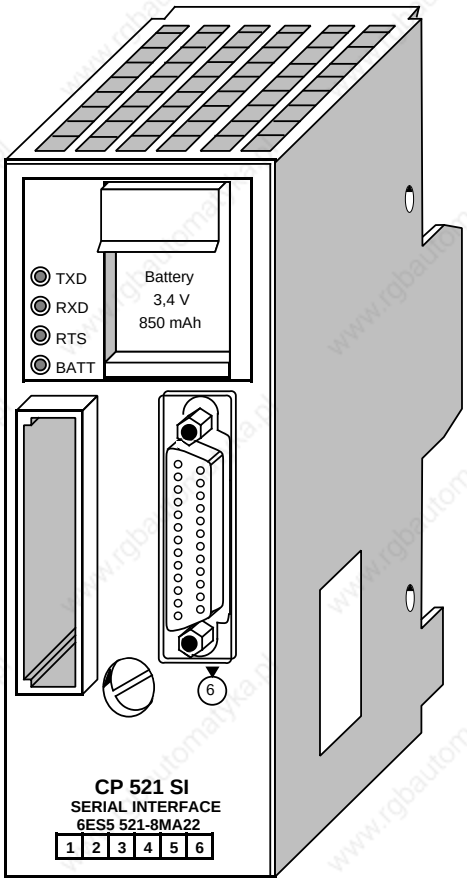
1. Plug the IP 267 into a bus unit like any other I/O module (see chapter 3).
2. Insert the IP 267 module only into slots 0 to 7.
3. Connect the external switches to the DIs of the IP 267 via the terminal block.
4. Connect the stepper motor's power section to the 9-pin subminiature D female connector.

**Addressing**

The IP 267 is addressed like an analog module.

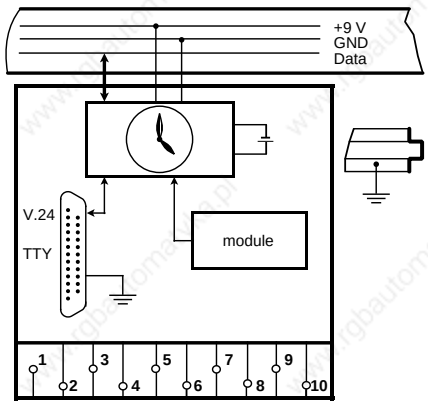
## 15.13 Communications Modules

### 15.13.1 Printer Communications Module CP 521SI (6ES5 521-8MA22)




**Technical Specifications**

| | |
|---|---|
| Galvanic isolation | TTY signals are isolated |
| Memory submodule | EPROM/EEPROM |
| Serial interface | V.24/TTY passive (active) |
| Transmission | Asynchronous 10-bit character frame/11-bit character frame |
| Transmission rate | 110 to 9600 baud |
| Permissible cable length<br>- V.24<br>- TTY | <br>15 m (49.2 ft.)<br>Results from: (Voltage drop on cable)+ (Receiver-typ. voltage drop 1.5 V) or (transmitter-typ. voltage drop 0.9 V)<br>max. 1000 m (3281 ft.) |
| LED displays<br>- TxD (green)<br>- RxD (green)<br>- RTS (green)<br>- BATT (yellow) | <br>Transmitting<br>Receiving<br>Ready to send<br>Battery failure |
| Back-up battery<br>Lithium 1/2 AA | <br>3.6 V/850 mAh |
| Current consumption from +9 V | typ. 140 mA |
| Power loss of module | typ. 1.2 W |
| Weight | approx. 500 g (1.1 lb.) |



The CP 521 SI (Serial Interface) communications module is a powerful I/O module with its own central processor.

A separate manual is available for this module. It can be ordered under the order number 6ES5 998-1UD21.

The following is an overview of the module's mode of operation.

**Function**

The CP 521 SI can be used for unidirectional and bidirectional data exchange.

**Unidirectional Data Exchange**

For unidirectional data traffic, the CP 521 is provided with a printer driver. If the printer driver is used, the following must be connected to the serial interface of the CP 521 SI:

- A printer with TTY interface (active) or
- A printer with V.24 (RS 232C) interface.

This enables you to log process states and process disturbances. The output of messages on the printer does not extend the response time of the programmable controller.

The following messages and texts can be output:

- Message texts, which you have configured on a memory submodule in data blocks DB 2 to 63.
- Time of day and date, which are provided by the module's own clock
- Values for variables which are transmitted to the CP 521 SI via the I/O bus.

The message texts are stored on an EPROM or EEPROM memory submodule (up to 8/16 Kbytes).

**Bidirectional Data Exchange**

The following drivers are implemented for the bidirectional data exchange:

- ASCII driver, transparent
- ASCII driver, interpreting mode I and interpreting mode II
- ”3964(R)” driver
- SINEC L1 driver, master (point-to-point)
- SINEC L1 driver, slave
- Terminal driver

The use of these drivers enables the transmission of data frames between the CPU and an I/O device connected to the CP 521 SI.

The maximum data flow rate is 6 bytes of user data per 2 program cycles; i.e. at a program cycle time of, for example, 50 ms a maximum of 60 bytes per second can be transmitted.

The following terminals and communications devices can be used as I/O devices:

- Keyboard
- Terminal
- Another CP 521 SI
- CP 523
- S5-95U with second serial interface
- CP 524/CP 525-2 (in connection with special driver 6ES5 897-2AB11)
- CPU 944 (with ASCII driver, 3964(R) driver)
- Other I/O devices with serial interface, e.g. bar code readers

Which of the I/O devices and transmission modes are used depends on the intended application of data transmission. In the bidirectional data exchange mode of the module you are, for example, able to network programmable controllers (point-to-point link).

I/O devices and CP 521 SI are connected with each other via a serial interface. Either a passive TTY interface or a 24 V voltage interface are available (programmable).

Parameterizing (matching) of the I/O interface and configuring of the message texts are supported by the DB editor of programmers. The parameters of the I/O interface are stored either on a memory submodule in DB1 or are directly transmitted in the user program. The CP 521 SI can be programmed and operated wihtout the COM software.

**Integrated Real-Time Clock**

The CP 521 SI has it's own real-time clock which is battery-backed when the module is in the de-energized state. Independent of the type of function selected for the CP 521 SI, the clock data can be read from the CPU and can be used in the user program for date and time-dependent tasks.

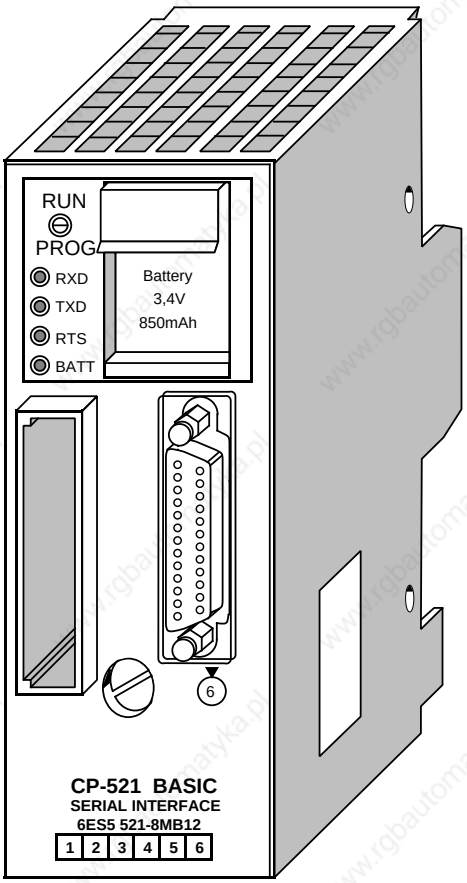
**Installation**

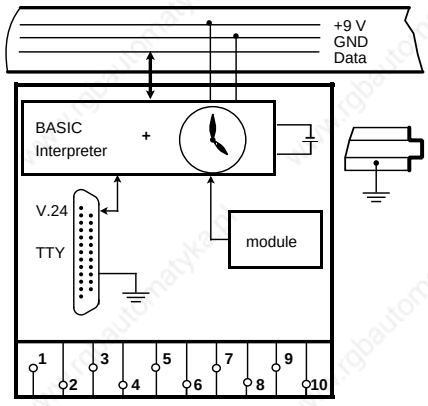
- As other I/O modules, the CP 521 SI is mounted on the bus unit (see chapter 3).
- Plug the module only into slots 0 to 7.
- The module has no connection to the terminal block.
- Connect the printer to the module via a 25-pin sub-D female connector.

**Adressing**

The CP 521 SI is addressed like a 4-channel analog module.

## 15.13.2 Communications Module CP 521 BASIC (6ES5 521-8MB12)




**Technical Specifications**

| | | |
|---|---|---|
| Galvanic isolation | | TTY signals are isolated |
| Memory submodule | | EPROM/EEPROM/ RAM |
| Serial interface | | V.24 (RS-232-C)/TTY, passive |
| Real-time clock<br>- accuracy | | ±1 s/day at 25 °C (77 °F) |
| - variation due to temperature change $t_V$ (ambient temperature $T_A$ in °C) | | -10 to +70 °C<br>1 s to -11 s according to data sheet |
| Quartz frequency | | 14.7456 MHz |
| Transmission mode | | asynchronous 10-bit character frame/11-bit character frame |
| Baud rate | | 110 to 9600 Bd |
| LEDs<br>- TXD<br>- RXD<br>- RTS<br>- BATT (yellow) | | <br>send data<br>receive data<br>ready to send<br>battery low |
| Permiss. length of cable<br>- TTY dependent on: voltage drop on the +line<br>- typical for receiver<br>- typical for sender<br>- V.24 | | <br><br>1.5 V+<br>0.9 V<br>15 m (50 ft.) |
| Back-up battery<br>lithium AA<br>Life expectancy | | <br>3.4 V/850 mAh<br>1 year minimum |
| Degree of protection | | IP20 |
| Permiss. ambient temperature<br>- horizontal arrangement<br>- vertical arrangement | | <br>0 to 60 °C (32 to 140 °F)<br>0 to 40 °C (32 to 104 °F) |
| Relative humidity | | 15% to 95% |
| Current consumption from+9 V (CPU) | typ. | 180 mA |
| Power loss of the module | typ. | 1.6 W |
| Weight | approx. | 500 g (1 lb. 1.5 oz.) |

Note:
It is only possible to run the CP 521 with the interrupt processing if the interrupts are disabled at the end of the OB1 cycle and enabled again at the beginning of the OB1 cycle.

The CP 521 BASIC is a powerful peripheral module that can be used with the SIMATIC systems S5-90U, S5-95U, and the S5-100U. It has its own central processor (cannot be used with the CPU 100, version 8MA01).

A separate manual for this module is available. The order number is 6ES5 998-0UW21.

A brief overview of the functions of this module follows.

**Function**

This module comes with a special COM software package that is required for generating and storing BASIC programs (on a floppy disk or an EPROM submodule).

Since the CP 521 includes a basic interpreter, you can create and run BASIC programs that exchange data with a CPU and a connected peripheral device. Use a programmer or a PC terminal and the COM software to program the BASIC interpreter.

You can store the BASIC programs in the module's own battery backed-up RAM or on a plug-in memory submodule.

Connect programmers or PC terminals to the CP 521 via a serial interface. You can choose (by setting parameters) between a passive TTY current-loop interface or a RS-232 C V.24 interface to connect a programmer or terminal. Connect a printer to the unidirectional V.24 interface of the module to print listings or messages.

Change parameter settings for the peripheral interface by using a BASIC command or by using the BASIC program.

The CP 521 has an integral real-time clock that can be backed up by a battery. You can use the clock data in unidirectional data traffic to log process statuses or process malfunctions.

**Installation**

1. Install the communications module on the bus module like any other I/O module (see chapter 3).
2. Plug the module only into slots 0 to 7.
3. The module has no connection to the terminal block.
4. Connect the printer to the module via a 25-pin sub-D female connector.

**Addressing**

The module is addressed like a 4-channel analog module.

| Appendices | |
|---|---|
| Appendix A | Operations List, Machine Code and List of Abbreviations |
| Appendix B | Dimension Drawings |
| Appendix C | Active and Passive Faults in Automation Equipment / Guidelines for Handling Electrostatic Sensitive Devices |
| Appendix D | Information for Ordering Accessories |
| Appendix E | Reference Materials |
| Appendix F | Siemens Addresses Worldwide |

# A Operations List, Machine Code and List of Abbreviations


| | | | |
|---|---|---|---|
| A.1 | Operations List | A - | 1 |
| A.1.1 | Basic Operations | A - | 1 |
| A.1.2 | Supplementary Operations | A - | 8 |
| A.1.3 | System Operations, for CPU 102 and Higher | A - | 13 |
| A.1.4 | Evaluation of CC 1 and CC 0 | A - | 14 |
| A.2 | Machine Code Listing | A - | 15 |
| A.3 | List of Abbreviations | A - | 18 |

# A Operations List, Machine Code and Abbreviations

## A.1 Operations List

### A.1.1 Basic Operations

☒ For organization blocks (OB) ☒ For function blocks (FB)
☒ For program blocks (PB) ☒ For sequence blocks (SB)

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs: CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Boolean Logic Operations** | | | | | | | | | |
| A | I, Q | N | Y | N | typ. 70 | 4 | 1.6 | 0.8 | Scan operand for “1” and combine with RLO through logic AND. |
| | F | N | Y | N | | | | | |
| | T | N | Y | N | | 7 | | | |
| | C | N | Y | N | | | | | |
| AN | I, Q | N | Y | N | typ. 75 | 4 | 1.6 | 0.8 | Scan operand for “0” and combine with RLO through logic AND. |
| | F | N | Y | N | | | | | |
| | T | N | Y | N | | 9 | | | |
| | Z | N | Y | N | | | | | |
| O | I, Q | N | Y | N | typ. 75 | 4 | 1.6 | 0.8 | Scan operand for “1” and combine with RLO through logic OR. |
| | F | N | Y | N | | | | | |
| | T | N | Y | N | | 7 | | | |
| | C | N | Y | N | | | | | |
| ON | I, Q | N | Y | N | typ. 80 | 4 | 1.6 | 0.8 | Scan operand for “0” and combine with RLO through logic OR. |
| | F | N | Y | N | | | | | |
| | T | N | Y | N | | 9 | | | |
| | C | N | Y | N | | | | | |
| O | | N | Y | Y | 41 | 7 | 1.6 | 0.8 | Combine AND operations through logic OR. |
| A( | | N | Y | Y | 61 | 6 | 1.6 | 0.8 | Combine expressions enclosed in parentheses through logic AND (6 nesting levels). |
| O( | | N | Y | Y | 64 | 6 | 1.6 | 0.8 | Combine expressions enclosed in parentheses through logic OR (6 nesting levels). |
| ) | | N | Y | N | 51 | 13 | 1.6 | 0.8 | Close parentheses (conclusion of a parenthetical expression). |
| **Set/Reset Operations** | | | | | | | | | |
| S | I, Q | Y | N | Y | typ. 70 | 7 | 1.6 | 0.8 | Set operand to “1”. |
| | F | Y | N | Y | | | | | |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | 2 | 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Set/Reset Operations (cont.)** | | | | | | | | | |
| R | I, O | Y | N | Y | typ. 70 | 7 | 1.6 | 0.8 | Reset operand to “0”. |
| | F | Y | N | Y | | | | | |
| = | I, O | N | N | Y | typ. 70 | 6 | 1.6 | 0.8 | Assign value of RLO to operand. |
| | F | N | N | Y | | | | | |
| **Load Operations** | | | | | | | | | |
| L | IB | N | N | N | 59 | 14 | 1.6 | 0.8 | Load an input byte from the PII into ACCU 1. |
| L | QB | N | N | N | 63 | 14 | 1.6 | 0.8 | Load an output byte from the PIQ into ACCU 1. |
| L | IW | N | N | N | 59 | 17 | 1.6 | 0.8 | Load an input word from the PII into ACCU 1: byte n ACCU 1 (bits 8-15); byte n+1 ACCU 1 (bits 0-7) |
| L | QW | N | N | N | 63 | 17 | 1.6 | 0.8 | Load an output word from the PIQ into ACCU 1: byte n ACCU 1 (bits 8-15); byte n+1 ACCU 1 (bits 0-7) |
| L | PY | -- | -- | N | -- | -- | 91 | 68 | Permissible in OB2 and OB13. Load an input byte of the digital/analog inputs from the interrupt PII into ACCU 1. |
| L | PW | -- | -- | N | -- | -- | 92 | 69 | Permissible in OB2 and OB13. Load an input byte of the digital/analog inputs from the interrupt PII into ACCU 1. |
| L | FY | N | N | N | 64 | 14 | 1.6 | 0.8 | Load a flag byte into ACCU 1. |
| L | FW | N | N | N | 71 | 17 | 1.6 | 0.8 | Load a flag word into ACCU 1: byte n ACCU 1 (bits 8-15); byte n+1 ACCU 1 (bits 0-7). |
| L | DL | N | N | N | 65 | 39 | 82 | 1.7 | Load a data word (left-hand byte) of the current data block into ACCU 1. |
| L | DR | N | N | N | 65 | 41 | 83 | 1.7 | Load a data word (right-hand byte) of the current data block into ACCU 1. |
| L | DW | N | N | N | 66 | 43 | 85 | 2.0 | Load a data word of the current data block into ACCU 1: byte n ACCU 1 (bits 8-15); byte n+1 ACCU 1 (bits 0-7). |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | 2 | 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Load Operations (cont.)** | | | | | | | | | |
| L | KB | N | N | N | 54 | 7 | 59 | 1.45 | Load a constant (1-byte number) into ACCU 1. |
| L | KS | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (2 characters in ASCII format) into ACCU 1. |
| L | KF | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (fixed-point number) into ACCU 1. |
| L | KH | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (hexadecimal code) into ACCU 1. |
| L | KM | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (bit pattern) into ACCU 1. |
| L | KY | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (2-byte number) into ACCU 1. |
| L | KT | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (time in BCD) into ACCU 1. |
| L | KC | N | N | N | 57 | 7 | 1.6 | 0.8 | Load a constant (count in BCD) into ACCU 1. |
| L | T, C | N | N | N | typ. 70 | 19 | 1.6 | 0.8 | Load a time or count (in binary code) into ACCU 1. |
| LC | T | N | N | N | 125 | 69 | 154 | 1.8 | Load times or counts (in BCD) into ACCU 1. |
| | C | N | N | N | | | | | |
| **Transfer Operations** | | | | | | | | | |
| T | IB | N | N | N | 51 | 5 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to an input byte (into the PII). |
| T | QB | N | N | N | 54 | 5 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to an output byte (into the PIQ). |
| T | IW | N | N | N | 53 | 11 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to an input word (into the PII): ACCU 1 (bits 8-15) byte n; ACCU 1 (bits 0-7) byte n+1. |
| T | QW | N | N | N | 56 | 11 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to an output word (into the PIQ): ACCU 1 (bits 8-15) byte n; ACCU 1 (bits 0-7) byte n+1. |
| T | PY | -- | -- | N | -- | -- | 60 | 37 | Permissible in OB2 and OB13. Transfer the contents of ACCU 1 to the interrupt PIQ with updating of the PIQ. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Transfer Operations (cont.)** | | | | | | | | | |
| T | PW | -- | -- | N | | | 67 | 51 | Permissible in OB2 and OB13. Transfer the contents of ACCU 1 to the interrupt PIQ with updating of the PIQ. |
| T | FY | N | N | N | 55 | 5 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to a flag byte. |
| T | FW | N | N | N | 64 | 11 | 1.6 | 0.8 | Transfer the contents of ACCU 1 to a flag word (into the PIQ): ACCU 1 (bits 8-15) byte n; ACCU 1 (bits 0-7) byte n+1. |
| T | DL | N | N | N | 53 | 31 | 75 | 1.15 | Transfer the contents of ACCU 1 to a data word (left-hand byte). |
| T | DR | N | N | N | 57 | 33 | 78 | 1.15 | Transfer the contents of ACCU 1 to a data word (right-hand byte). |
| T | DW | N | N | N | 59 | 36 | 81 | 1.4 | Transfer the contents of ACCU 1 to a data word. |
| **Timer Operations** | | | | | | | | | |
| SP | T | Y | N | Y | 125 | 74 | 147 | 1.9 | Start a timer (stored in ACCU 1) as a signal-contracting pulse. |
| SE | T | Y | N | Y | 125 | 74 | 147 | 1.9 | Start a timer (stored in ACCU 1) as extended pulse (signal contracting and stretching). |
| SD | T | Y | N | Y | 127 | 76 | 150 | 1.9 | Start an on-delay timer (stored in ACCU 1). |
| SS | T | Y | N | Y | 127 | 76 | 150 | 1.9 | Start a stored on-delay timer (stored in ACCU 1). |
| SF | T | Y | N | Y | 125 | 74 | 144 | 1.9 | Start an off-delay timer (stored in ACCU 1). |
| R | T | Y | N | Y | 126 | 75 | 96 | 1.9 | Reset a timer. |
| **Counter Operations** | | | | | | | | | |
| CU | C | Y | N | Y | 79 | 42 | 105 | 1.9 | Counter counts up 1. |
| CD | C | Y | N | Y | 92 | 31 | 117 | 1.9 | Counter counts down 1. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Counter Operations (cont.)** | | | | | | | | | |
| S | C | Y | N | Y | 118 | 67 | 141 | 1.9 | Set counter. |
| R | C | Y | N | Y | 69 | 12 | 96 | 1.9 | Reset counter. |
| **Arithmetic Operations** | | | | | | | | | |
| +F | | N | N | N | 55 | 26 | 1.6 | 0.8 | Add two fixed-point numbers: ACCU 1+ACCU 2. CC 1/CC 0/OV are affected. |
| -F | | N | N | N | 58 | 23 | 1.6 | 0.8 | Subtract one fixed-point number from another: ACCU 2 - ACCU 1. CC 1/CC 0/OV are affected. |
| **Comparison Operations** | | | | | | | | | |
| !=F | | N | Y | N | 79 | 24 | 1.6 | 0.8 | Compare two fixed-point numbers for “equal to”: If ACCU 2=ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |
| ><F | | N | Y | N | 82 | 27 | 1.6 | 0.8 | Compare two fixed-point numbers for “not equal to”: If ACCU 2 ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |
| >F | | N | Y | N | 79 | 24 | 1.6 | 0.8 | Compare two fixed-point numbers for “greater than”: If ACCU 2 > ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |
| >=F | | N | Y | N | 79 | 24 | 1.6 | 0.8 | Compare two fixed-point numbers for “greater than or equal to”: If ACCU 2 ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |
| <F | | N | Y | N | 82 | 27 | 1.6 | 0.8 | Compare two fixed-point numbers for “less than”: If ACCU 2 < ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |
| <=F | | N | Y | N | 82 | 27 | 1.6 | 0.8 | Compare two fixed-point numbers for “less than or equal to”: If ACCU 2 ACCU 1, the RLO is “1”. CC 1/CC 0 are affected. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Block Call Operations** | | | | | | | | | |
| JU | PB | N | N | Y | 125 | 49 | 185 | 3.35 | Jump unconditionally to a program block. |
| JU | FB | N | N | Y | 147 | 49 | 187 | 3.35 | Jump unconditionally to a function block. |
| JU | SB | N | N | Y | -- | -- | 185 | 3.35 | Jump unconditionally to a sequence block. |
| JC | PB | Y | Y[1)] | Y | 130 | 53 | 190 | 3.35 | Jump conditionally to a program block. |
| JC | FB | Y | Y[1)] | Y | 152 | 53 | 196 | 3.35 | Jump conditionally to a function block. |
| JC | SB | Y | Y[1)] | Y | -- | -- | 194 | 3.35 | Jump conditionally to a sequence block. |
| C | DB | N | N | N | 70 | 28 | 79 | 1.75 | Call a data block. |
| G | DB | N | N | Y | -- | -- | 233 | 182 | Generate or delete a data block. |
| **Return Operations** | | | | | | | | | |
| BE | | N | N | Y | 88 | 36 | 119 | 2.5 | Block end (termination of a block) |
| BEC | | Y | Y[1)] | Y | 90 | 38 | 121 | 2.5 | Block end, conditional |
| BEU | | N | N | Y | 88 | 36 | 119 | 2.5 | Block end, unconditional (BEU cannot be used in organization blocks.) |
| **“No” Operations** | | | | | | | | | |
| NOP 0 | | N | N | N | 35 | 0 | 1.6 | 0.8 | No operation (all bits reset) |
| NOP 1 | | N | N | N | 35 | 0 | 1.6 | 0.8 | No operation (all bits set) |
| **Stop Operations** | | | | | | | | | |
| STP | | N | N | N | 35 | 1 | 53 | 25 | Stop: scanning is still completed before a stop. Error ID “STS” is set in the ISTACK. |
| **Display Generation Operations** | | | | | | | | | |
| BLD 130 | | N | N | N | 35 | 0 | 1.6 | 0.8 | Display generation operation for the programmer: carriage return generates blank line. |
| BLD 131 | | N | N | N | 35 | 0 | 1.6 | 0.8 | Display generation operation for the programmer: switch to statement list (STL). |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

1) RLO is set to “1”.

| Operation (STL) | Permissible Operands | RLO* 1 | 2 | 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Display Generation Operations (cont.)** | | | | | | | | | |
| BLD 132 | | N | N | N | 35 | 0 | 1.6 | 0.8 | Display generation operation for the programmer: switch to control system flowchart (CSF). |
| BLD 133 | | N | N | N | 35 | 0 | 1.6 | 0.8 | Display generation operation for the programmer: switch to ladder diagram (LAD). |
| BLD 255 | | N | N | N | 35 | 0 | 1.6 | 0.8 | Display generation operation for the programmer: terminate a segment. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

## A.1.2 Supplementary Operations

☐ For organization blocks (OB) ☒ For function blocks (FB)
☐ For program blocks (PB) ☐ For sequence blocks (SB)

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs: CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Boolean Logic Operations** | | | | | | | | | |
| A= | Formal-operand I, Q, F, T, C | N | Y | N | -- | -- | 202 | 151 | AND operation: scan formal operand for “1”. (Data type: BI) |
| AN= | Formal operand I, Q, F, T, C | N | Y | N | -- | -- | 202 | 151 | AND operation: scan formal operand for “0”. (Data type: BI) |
| O= | Formal operand I, Q, F, T, C | N | Y | N | -- | -- | 202 | 151 | OR operation: scan formal operand for “1”. (Data type: BI) |
| ON= | Formal operand I, Q, F, T, C | N | Y | N | -- | -- | 202 | 151 | OR operation: scan formal operand for “0”. (Data type: BI) |
| AW | | N | N | N | 53 | 19 | 1.6 | 0.8 | Combine contents of ACCU 2 and ACCU 1 through logic AND (word operation). Result is stored in ACCU 1. CC 1/CC 0 are affected. |
| OW | | N | N | N | 53 | 19 | 1.6 | 0.8 | Combine contents of ACCU 2 and ACCU 1 through logic OR (word operation). Result is stored in ACCU 1. CC 1/CC 0 are affected. |
| XOW | | N | N | N | 51 | 19 | 1.6 | 0.8 | Combine contents of ACCU 2 and ACCU 1 through logic EXCLUSIVE OR (word operation). Result is stored in ACCU 1. CC 1/CC 0 are affected. |
| **Bit Operations** | | | | | | | | | |
| TB | T, C | N | Y | N | -- | -- | 187 | 123 | Test a bit of a timer or counter word for “1”. |
| TB | D | N | Y | N | -- | -- | 187 | 144 | Test a bit of a data word for “1”. |
| TB | RS | N | Y | N | -- | -- | 185 | 121 | Test a bit of a data word in the system data area for “1”. |
| TBN | T, C | N | Y | N | -- | -- | 188 | 124 | Test a bit of a timer or counter word for “0”. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs: CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Bit Operations (cont.)** | | | | | | | | | |
| TBN | D | N | Y | N | -- | -- | 188 | 145 | Test a bit of a data word for “0”. |
| TBN | RS | N | Y | N | -- | -- | 186 | 122 | Test a bit of a data word in the system data area for “0”. |
| SU | T, C | N | N | Y | -- | -- | 180 | 125 | Set a bit of a timer or counter word unconditionally. |
| SU | D | N | N | Y | -- | -- | 183 | 146 | Set a bit of a data word unconditionally. |
| RU | T, C | N | N | Y | -- | -- | 189 | 124 | Reset a bit of a timer or counter word unconditionally. |
| RU | D | N | N | Y | -- | -- | 189 | 146 | Reset a bit of a data word unconditionally. |
| **Set/Reset Operations** | | | | | | | | | |
| S= | Formal operand I, Q, F | Y | N | Y | -- | -- | 202 | 151 | Set a formal operand (when RLO=1). (Data type: BI) |
| RB= | Formal operand I, Q, F | Y | N | Y | -- | -- | 203 | 152 | Reset a formal operand (when RLO=1). (Data type: BI) |
| RD= | Formal operand T, C | Y | N | Y | -- | -- | 197 | 147 | Reset a formal operand (digital) (when RLO=1). |
| == | Formal operand I, Q, F | N | N | Y | -- | -- | 202 | 151 | Assign the value of the RLO to the status of the formal operand. (Data type: BI) |
| FR | T, C | Y | N | Y | -- | -- | 98 | 1.9 | Enable a timer/counter for cold restart. If RLO=“1”,<br>- “FR T” restarts the timer<br>- “FR C” sets, decrements, or increments the counter. |
| FR= | Formal op. T, C | Y | N | Y | -- | -- | 194** | 145** | Enable formal operand (timer/counter) for cold restart (for detailed description, see “FR” operation). |
| SP= | Formal op. T | Y | N | Y | -- | -- | 194** | 145** | Start a timer (formal operand) as pulse with the value stored in ACCU 1. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

** +Execution of the substituted command

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Timer and Counter Operations (cont.)** | | | | | | | | | |
| SD= | Formal op. T | Y | N | Y | -- | -- | 194** | 145** | Start an on-delay timer (formal operand) with the value stored in ACCU 1. |
| SEC = | Formal op. T, C | Y | N | Y | -- | -- | 194** | 145** | Start a timer (formal operand) as an extended pulse with the value stored in ACCU 1, or set a counter (formal operand) with the next count value indicated. |
| SSU = | Formal op. T, C | Y | N | Y | -- | -- | 194** | 145** | Start a stored on-delay timer (formal operand) with the value stored in ACCU 1, or increment a counter (formal operand). |
| SFD= | Formal op. T, C | Y | N | Y | -- | -- | 194** | 145** | Start an off-delay timer (formal operand) with the value stored in ACCU 1, or decrement a counter (formal operand). |
| **Load and Transfer Operations** | | | | | | | | | |
| L= | Formal operand I, Q, F, T, C | N | N | N | -- | -- | 142** | 148** | Load the value of the formal operand into ACCU 1.<br>Data type: BY, W<br>Additional actual operands: DL, DR, DW |
| L | RS | N | N | N | -- | -- | 77 | 61 | Load a word from the system data area into ACCU 1. |
| LD= | Formal operand T, C | N | N | N | -- | -- | 194** | 145** | Load the value of the formal operand in BCD code into ACCU 1. |
| LW= | Formal operand | N | N | N | -- | -- | 152 | 76 | Load a formal operand bit pattern into ACCU 1.<br>Data type: D<br>Parameter type:<br>KC, KF, KH, KM, KS, KT, KY |
| T= | Formal operand I, Q, F | N | N | N | -- | -- | 195** | 149** | Transfer the contents of ACCU 1 to the formal operand.<br>Data type: BY, W<br>Additional actual operands: DR, DL, DW |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

** +Processing time for the substituted command

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs: CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Conversion Operations** | | | | | | | | | |
| CFW | | N | N | N | 42 | 4 | 1.6 | 0.8 | Form the one's complement of ACCU 1. |
| CSW | | N | N | N | 60 | 23 | 1.6 | 0.8 | Form the two's complement of ACCU 1. CC 1/CC 0 and OV are affected. |
| **Shift Operations** | | | | | | | | | |
| SLW | Parameter n=0 to 15 | N | N | N | 47+ n·10 | 12+ n·10 | 1.6 | 0.8 | Shift the contents of ACCU 1 to the left by the value specified in the parameter. Unassigned positions are padded with zeros. CC 1/CC 0 are affected. |
| SRW | Parameter n=0 to 15 | N | N | N | 47+ n·10 | 12+ n·10 | 1.6 | 0.8 | Shift the contents of ACCU 1 to the right by the value specified in the parameter. Unassigned positions are padded with zeros. CC 1/CC 0 are affected. |
| **Jump Operations** | | | | | | | | | |
| JU= | Symbolic address max. 4 characters | N | N | N | 62 | 2 | 1.6 | 0.8 | Jump unconditionally to the symbolic address. |
| JC= | Symbolic address max. 4 characters | Y | Y[1)] | Y | 65 | 5 | 1.6 | 0.8 | Jump conditionally to the symbolic address. (If the RLO is “0”, it is set to “1”.) |
| JZ= | Symbolic address max. 4 characters | N | N | N | 69 | 6 | 1.6 | 0.8 | Jump if the result is zero. The jump is made only if CC 1=0 and CC 0=0. The RLO is not changed. |
| JN= | Symbolic address max. 4 characters | N | N | N | 69 | 10 | 1.6 | 0.8 | Jump if the result is not zero. The jump is made only if CC 1 CC 0. The RLO is not changed. |
| JP= | Symbolic address max. 4 characters | N | N | N | 71 | 6 | 1.6 | 0.8 | Jump if the sign of the result is “+”. The jump is made only if CC 1=1 and CC 0=0. The RLO is not changed. |
| JM= | Symbolic address max. 4 characters | N | N | N | 71 | 6 | 1.6 | 0.8 | Jump if the sign of the result is “-”. The jump is made only if CC 1=0 and CC 0=1. The RLO is not changed. |
| JO= | Symbolic address max. 4 characters | N | N | N | 65 | 4 | 1.6 | 0.8 | Jump on overflow. The jump is made only if the OVERFLOW bit is set. The RLO is not changed. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

1) RLO is set to “1”.

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs: CPU 100 | Execution Time in µs: CPU 102 | Execution Time in µs: CPU 103 MA02 | Execution Time in µs: CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Other Operations** | | | | | | | | | |
| IA | | N | N | N | -- | -- | 58 | 24 | Disable interrupt. Input/ output interrupt or timer OB processing is disabled. |
| RA | | N | N | N | -- | -- | 58 | 26 | Enable interrupt. This operation cancels the effect of IA. |
| D | | N | N | N | -- | -- | 49 | 0,9 | Decrement the low byte (bits 0 to 7) of ACCU 1 by the value n (n=0 to 255). |
| I | | N | N | N | -- | -- | 49 | 0,9 | Increment the low byte (bits 0 to 7) of ACCU 1 by the value n (n=0 to 255). |
| DO= | Formal operand | N | N | Y | -- | -- | 252** | 188** | Process a block. (Only C DB, JU OB, J U PB, JU FB, JU SB can be substituted.) Actual operands: C DB, JU OB, JU PB, JU FB, JU SB |
| DO | DW*** | N | N | N | -- | -- | 229 | 171 | Process data word. The next operation is combined with the parameter specified in the data word (OR operation) and then carried out. |
| DO | FW*** | N | N | N | -- | -- | 179 | 138 | Process flag word. The next operation is combined with the parameter specified in the flag word (OR operation) and then carried out. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

** +Processing time for the substituted command

*** Permissible operations:

| | |
|---|---|
| A, AN, O, ON | L, LC, T; |
| S, R=; | JU, JC, JZ, JN, JP, JM, JO, SLW, SRW; |
| FR T, RT, SF T, SD T, SP T, SS T, SE T; | D, I; |
| FR C, RC, SC, CU, CD C; | C DB, T RS, TNB |

## A.1.3 System Operations, for CPU 102 and Higher

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Set Operations** | | | | | | | | | |
| SU | RS | N | N | Y | -- | -- | 167 | 123 | Set bit in system data area unconditionally. |
| RU | RS | N | N | Y | -- | -- | 167 | 123 | Reset bit in system data area unconditionally. |
| **Load and Transfer Operations** | | | | | | | | | |
| LIR | | N | N | N | -- | -- | 105 | 76 | Load the contents of a memory word (addressed by ACCU 1) indirectly into the register (0: ACCU 1; 2: ACCU 2). |
| TIR | | N | N | N | -- | -- | 85 | 61 | Transfer the register contents (0: ACCU 1; 2: ACCU 2) indirectly into the memory word (addressed by ACCU 1). |
| TNB | Parameter n=0 to 255 | N | N | N | -- | 13+n·19 (48+n·19) | 97+ n·21 | 75+ n·16 | Transfer a field byte by byte (number of bytes 0 to 255). |
| T | RS | N | N | N | -- | -- | 71 | 59 | Transfer a word to the system data area. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

| Operation (STL) | Permissible Operands | RLO* 1 | RLO* 2 | RLO* 3 | Execution Time in µs CPU 100 | CPU 102 | CPU 103 MA02 | CPU 103 MA03 | Function |
|---|---|---|---|---|---|---|---|---|---|
| **Block Call Operations and Return Operations** | | | | | | | | | |
| JU | OB | N | N | Y | -- | -- | 187 | 3.35 | Call an organization block unconditionally. |
| JC | OB | Y | Y[1)] | Y | -- | -- | 194 | 3.35 | Call an organization block conditionally. |
| **Jump Operation** | | | | | | | | | |
| JUR | | N | N | N | -- | -- | 131 | 82 | Jump at random within a function block (jump distance -32768 to + 32767) |
| **Arithmetic Operations** | | | | | | | | | |
| ADD | BF | N | N | N | -- | -- | 58 | 35 | Add byte constant (fixed point) to ACCU 1. |
| ADD | KF | N | N | N | -- | -- | 104 | 68 | Add fixed-point constant (word) to ACCU 1. |
| **Other Operations** | | | | | | | | | |
| STS | | N | N | N | -- | -- | | | Stop operation. Program processing is interrupted immediately after this operation. |
| TAK | | N | N | N | -- | -- | 74 | 57 | Swap the contents of ACCU 1 and ACCU 2. |

* 1 RLO dependent ? 2 RLO affected ? 3 RLO reloaded ?

1) RLO is set to “1”

## A.1.4 Evaluation of CC 1 and CC 0

| CC 1 | CC 0 | Arithmetic Operations | Digital Logic Operations | Comparison Operations | Shift Operations | Conversion Operations |
|---|---|---|---|---|---|---|
| 0 | 0 | Result = 0 | Result = 0 | ACCU 2 = ACCU 1 | shifted bit = 0 | — |
| 0 | 1 | Result < 0 | — | ACCU 2 < ACCU 1 | — | Result < 0 |
| 1 | 0 | Result > 0 | Result 0 | ACCU 2 > ACCU 1 | shifted bit = 1 | Result > 0 |

## A.2 Machine Code Listing

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| 0 | 0 | 0 | 0 | | | | | NOP 0 | |
| 0 | 1 | 0 | 0 | | | | | CFW | |
| 0 | 2 | $0_d$ | $0_d$ | | | | | L | T |
| 0 | 3 | $0_l$ | $0_l$ | | | | | TNB | |
| 0 | 4 | $0_d$ | $0_d$ | | | | | FR | T |
| 0 | 5 | 0 | 0 | | | | | BEC | |
| 0 | 6 | $0_c$ | $0_c$ | | | | | FR= | |
| 0 | 7 | $0_c$ | $0_c$ | | | | | A= | |
| 0 | 8 | 0 | 0 | | | | | IA | |
| 0 | 8 | 8 | 0 | | | | | RA | |
| 0 | 9 | 0 | 0 | | | | | CSW | |
| 0 | A | $0_a$ | $0_a$ | | | | | L | FY |
| 0 | B | $0_a$ | $0_a$ | | | | | T | FY |
| 0 | C | $0_d$ | $0_d$ | | | | | LC | T |
| 0 | D | $0_i$ | $0_i$ | | | | | JO= | |
| 0 | E | $0_c$ | $0_c$ | | | | | LC= | |
| 0 | F | $0_c$ | $0_c$ | | | | | 0 | |
| 1 | 0 | 8 | 2 | | | | | BLD | 130 |
| 1 | 0 | 8 | 3 | | | | | BLD | 131 |
| 1 | 0 | 8 | 4 | | | | | BLD | 132 |
| 1 | 0 | 8 | 5 | | | | | BLD | 133 |
| 1 | 0 | F | F | | | | | BLD | 255 |
| 1 | 1 | $0_n$ | $0_n$ | | | | | I | |
| 1 | 2 | $0_a$ | $0_a$ | | | | | L | FW |
| 1 | 3 | $0_a$ | $0_a$ | | | | | T | FW |
| 1 | 4 | $0_d$ | $0_d$ | | | | | SF | T |
| 1 | 5 | $0_i$ | $0_i$ | | | | | JP= | |
| 1 | 6 | $0_c$ | $0_c$ | | | | | SFD= | |
| 1 | 7 | $0_c$ | $0_c$ | | | | | S= | |
| 1 | 9 | $0_n$ | $0_n$ | | | | | D | |
| 1 | C | $0_d$ | $0_d$ | | | | | SE | T |
| 1 | D | $0_f$ | $0_f$ | | | | | JC | FB |

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| 1 | E | $0_c$ | $0_c$ | | | | | SEC= | |
| 1 | F | $0_c$ | $0_c$ | | | | | == | |
| 2 | 0 | $0_f$ | $0_f$ | | | | | C | DB |
| 2 | 1 | 2 | 0 | | | | | >F | |
| 2 | 1 | 4 | 0 | | | | | <F | |
| 2 | 1 | 6 | 0 | | | | | ><F | |
| 2 | 1 | 8 | 0 | | | | | !=F | |
| 2 | 1 | A | 0 | | | | | >=F | |
| 2 | 1 | C | 0 | | | | | <=F | |
| 2 | 2 | $0_g$ | $0_g$ | | | | | L | DL |
| 2 | 3 | $0_g$ | $0_g$ | | | | | T | DL |
| 2 | 4 | $0_d$ | $0_d$ | | | | | SD | T |
| 2 | 5 | $0_i$ | $0_i$ | | | | | JM= | |
| 2 | 6 | $0_c$ | $0_c$ | | | | | SD= | |
| 2 | 7 | $0_c$ | $0_c$ | | | | | AN= | |
| 2 | 8 | $0_e$ | $0_e$ | | | | | L | KB |
| 2 | A | $0_g$ | $0_g$ | | | | | L | DR |
| 2 | B | $0_g$ | $0_g$ | | | | | T | DR |
| 2 | C | $0_d$ | $0_d$ | | | | | SS | T |
| 2 | D | $0_i$ | $0_i$ | | | | | JU= | |
| 2 | E | $0_c$ | $0_c$ | | | | | SSU= | |
| 2 | F | $0_c$ | $0_c$ | | | | | ON= | |
| 3 | 0 | 0 | 1 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KC |
| 3 | 0 | 0 | 2 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KT |
| 3 | 0 | 0 | 4 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KF |
| 3 | 0 | 1 | 0 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KS |
| 3 | 0 | 2 | 0 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KY |
| 3 | 0 | 4 | 0 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KH |
| 3 | 0 | 8 | 0 | $0_e$ | $0_e$ | $0_e$ | $0_e$ | L | KM |
| 3 | 2 | $0_g$ | $0_g$ | | | | | L | DW |
| 3 | 3 | $0_g$ | $0_g$ | | | | | T | DW |
| 3 | 4 | $0_d$ | $0_d$ | | | | | SP | T |

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| 3 | 5 | $0_i$ | $0_i$ | | | | | JN= | |
| 3 | 6 | $0_c$ | $0_c$ | | | | | SI= | |
| 3 | 7 | $0_c$ | $0_c$ | | | | | RB= | |
| 3 | C | $0_d$ | $0_d$ | | | | | R | T |
| 3 | D | $0_f$ | $0_f$ | | | | | JU | FY |
| 3 | E | $0_c$ | $0_c$ | | | | | RD= | |
| 3 | F | $0_c$ | $0_c$ | | | | | LW= | |
| 4 | 0 | 0 | $0_k$ | | | | | LIR | |
| 4 | 1 | 0 | 0 | | | | | AW | |
| 4 | 2 | $0_o$ | $0_o$ | | | | | L | C |
| 4 | 4 | $0_o$ | $0_o$ | | | | | FR | C |
| 4 | 5 | $0_i$ | $0_i$ | | | | | JZ= | |
| 4 | 6 | $0_c$ | $0_c$ | | | | | L= | |
| 4 | 8 | 0 | $0_k$ | | | | | TIR | |
| 4 | 9 | 0 | 0 | | | | | OW | |
| 4 | A | $0_a$ | $0_a$ | | | | | L | IB |
| 4 | A | $8_a$ | $0_a$ | | | | | L | QB |
| 4 | B | $0_a$ | $0_a$ | | | | | T | IB |
| 4 | B | $8_a$ | $0_a$ | | | | | T | QB |
| 4 | C | $0_o$ | $0_o$ | | | | | LC | C |
| 4 | D | $0_f$ | $0_f$ | | | | | JC | OB |
| 4 | E | $0_g$ | $0_g$ | | | | | DO | FW |
| 5 | 0 | $0_e$ | $0_e$ | | | | | ADD | BF |
| 5 | 1 | 0 | 0 | | | | | XOW | |
| 5 | 2 | $0_a$ | $0_a$ | | | | | L | IW |
| 5 | 2 | $8_a$ | $0_a$ | | | | | L | QW |
| 5 | 3 | $0_a$ | $0_a$ | | | | | T | IW |
| 5 | 3 | $8_a$ | $0_a$ | | | | | T | QW |
| 5 | 4 | $0_o$ | $0_o$ | | | | | CD | C |
| 5 | 5 | $0_f$ | $0_f$ | $0_e$ | $0_e$ | $0_e$ | $0_e$ | JC | PB |
| 5 | 8 | 0 | 0 | | | | | ADD | KF |
| 5 | 9 | 0 | 0 | | | | | -F | |

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| 5 | C | $0_o$ | $0_o$ | | | | | S | C |
| 5 | D | $0_f$ | $0_f$ | | | | | JC | SB |
| 6 | 1 | $0_h$ | $0_h$ | | | | | SLW | |
| 6 | 2 | $0_g$ | $0_g$ | | | | | L | RS |
| 6 | 3 | $0_g$ | $0_g$ | | | | | T | RS |
| 6 | 5 | 0 | 0 | | | | | BE | |
| 6 | 5 | 0 | 1 | | | | | BEU | |
| 6 | 6 | $0_c$ | $0_c$ | | | | | T= | |
| 6 | 9 | $0_h$ | $0_h$ | | | | | SRW | |
| 6 | C | $0_o$ | $0_o$ | | | | | CU | C |
| 6 | D | $0_f$ | $0_f$ | | | | | JU | OB |
| 6 | E | $0_g$ | $0_g$ | | | | | DO | DW |
| 7 | 0 | 0 | 0 | | | | | STS | |
| 7 | 0 | 0 | 2 | | | | | TAK | |
| 7 | 0 | 0 | 3 | C | 0 | $0_o$ | $0_o$ | STP | |
| 7 | 0 | 1 | 5 | 8 | 0 | $0_o$ | $0_o$ | TB | C |
| 7 | 0 | 1 | 5 | 4 | 0 | $0_o$ | $0_o$ | TBN | C |
| 7 | 0 | 1 | 5 | 0 | 0 | $0_o$ | $0_o$ | SU | C |
| 7 | 0 | 1 | 5 | C | 0 | $0_d$ | $0_d$ | RU | C |
| 7 | 0 | 2 | 5 | 8 | 0 | $0_d$ | $0_d$ | TB | T |
| 7 | 0 | 2 | 5 | 4 | 0 | $0_d$ | $0_d$ | TBN | T |
| 7 | 0 | 2 | 5 | 0 | 0 | $0_d$ | $0_d$ | SU | T |
| 7 | 0 | 2 | 5 | C | $0_b$ | $0_g$ | $0_g$ | RU | T |
| 7 | 0 | 4 | 6 | 8 | $0_b$ | $0_g$ | $0_g$ | TB | D |
| 7 | 0 | 4 | 6 | 4 | $0_b$ | $0_g$ | $0_g$ | TBN | D |
| 7 | 0 | 4 | 6 | 0 | $0_b$ | $0_g$ | $0_g$ | SU | D |
| 7 | 0 | 4 | 6 | C | $0_b$ | $0_g$ | $0_g$ | RU | D |
| 7 | 0 | 5 | 7 | 8 | $0_b$ | $0_g$ | $0_g$ | TB | RS |
| 7 | 0 | 5 | 7 | 4 | $0_b$ | $0_g$ | $0_g$ | TBN | RS |
| 7 | 0 | 5 | 7 | 0 | $0_b$ | $0_g$ | $0_g$ | SU | RS |
| 7 | 0 | 5 | 7 | | | | | RU | RS |

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| 7 | 2 | $0_a$ | $0_a$ | | | | | L | PY |
| 7 | 3 | $0_a$ | $0_a$ | | | | | T | PY |
| 7 | 5 | $0_f$ | $0_f$ | | | | | JU | PY |
| 7 | 6 | $0_c$ | $0_c$ | | | | | DO= | |
| 7 | 8 | 0 | 5 | 0 | 0 | $0_f$ | $0_f$ | G | DB |
| 7 | 9 | 0 | 0 | | | | | +F | |
| 7 | A | $0_a$ | $0_a$ | | | | | L | PW |
| 7 | B | $0_a$ | $0_a$ | | | | | T | PW |
| 7 | C | $0_o$ | $0_o$ | | | | | R | C |
| 7 | D | $0_f$ | $0_f$ | | | | | JU | SB |
| 7 | E | 0 | 0 | | | | | DI | |
| 8 | $0_b$ | $0_a$ | $0_a$ | | | | | A | F |
| 8 | $8_b$ | $0_a$ | $0_a$ | | | | | O | F |
| 9 | $0_b$ | $0_a$ | $0_a$ | | | | | S | F |
| 9 | $8_b$ | $0_a$ | $0_a$ | | | | | = | F |
| A | $0_b$ | $0_a$ | $0_a$ | | | | | AN | F |
| A | $8_b$ | $0_a$ | $0_a$ | | | | | ON | F |
| B | $0_b$ | $0_a$ | $0_a$ | | | | | R | F |
| B | 8 | $0_o$ | $0_o$ | | | | | A | C |
| B | 9 | $0_o$ | $0_o$ | | | | | O | C |
| B | A | 0 | 0 | | | | | A( | |
| B | B | 0 | 0 | | | | | O( | |
| B | C | $0_o$ | $0_o$ | | | | | AN | C |
| B | D | $0_o$ | $0_o$ | | | | | ON | C |

| Machine Code | | | | | | | | Operation | Operand |
|---|---|---|---|---|---|---|---|---|---|
| B0 | | B1 | | B2 | | B3 | | | |
| L | R | L | R | L | R | L | R | | |
| B | F | 0 | 0 | | | | | ) | |
| C | $0_b$ | $0_a$ | $0_a$ | | | | | A | I |
| C | $0_b$ | $8_a$ | $0_a$ | | | | | A | Q |
| C | $8_b$ | $0_a$ | $0_a$ | | | | | O | I |
| C | $8_b$ | $8_a$ | $0_a$ | | | | | O | Q |
| D | $0_b$ | $0_a$ | $0_a$ | | | | | S | I |
| D | $0_b$ | $8_a$ | $0_a$ | | | | | S | Q |
| D | $8_b$ | $0_a$ | $0_a$ | | | | | = | I |
| D | $8_b$ | $8_a$ | $0_a$ | | | | | = | Q |
| E | $0_b$ | $0_a$ | $0_a$ | | | | | AN | I |
| E | $0_b$ | $8_a$ | $0_a$ | | | | | AN | Q |
| E | $8_b$ | $0_a$ | $0_a$ | | | | | ON | I |
| E | $8_b$ | $8_a$ | $0_a$ | | | | | ON | Q |
| F | $0_b$ | $0_a$ | $0_a$ | | | | | R | I |
| F | $0_b$ | $8_a$ | $0_a$ | | | | | R | Q |
| F | 8 | $0_d$ | $0_d$ | | | | | A | T |
| F | 9 | $0_d$ | $0_d$ | | | | | O | T |
| F | A | $0_i$ | $0_i$ | | | | | JC= | |
| F | B | 0 | 0 | | | | | O | |
| F | C | $0_d$ | $0_d$ | | | | | AN | T |
| F | D | $0_d$ | $0_d$ | | | | | ON | T |
| F | F | F | F | | | | | NOP 1 | |

**Explanation of the Indices**

**a + byte address**
**b + bit address**
**c + parameter address**
**d + timer number**
**e + constant**
**f + block number**
**g + word address**
**h + number of shifts**
**i + relative jump address**
**k + register address**
**l + block length in bytes**
**m + jump displacement (16 bits)**
**n + value**
**o + counter number**

## A.3 List of Abbreviations

| Abbreviation | Explanation | Permissible Operand Value Range for | | |
|---|---|---|---|---|
| | | CPU 100 | CPU 102 | CPU 103 |
| ACCU 1 | Accumulator 1 (When accumulator 1 is loaded, any existing contents are shifted into accumulator 2.) | | | |
| ACCU 2 | Accumulator 2 | | | |
| BF | Byte constant (fixed-point number) | (- 127 to +127) | (- 127 to +127) | (- 127 to +127) |
| C | Counter<br>- retentive<br>- non-retentive<br>- for the “Bit Test” and “Set” supplementary operations | (0 to 7)<br>(8 to 15)<br>(0 to 15) | (0 to 7)<br>(8 to 127)<br>(0 to 127) | (0 to 7)<br>(8 to 127)<br>(0 to 127)<br>(0.0 to 127.15) |
| CC 0/CC 1 | Condition code 0/Condition code 1 | | | |
| CF | DB1 parameter: input correction factor (integral real-time clock) | | | |
| CLK | DB1 parameter: clock data location | | | |
| CPU | Central processing unit of programmable controller | | | |
| CSF | STEP 5 control system flowchart method of representation | | | |
| D | Data (1 bit) | | | (0.0 to 255.15) |
| DB | Data block | (2 to 63) | (2 to 63) | (2 to 255) |
| DL | Data word (left-hand byte) | (0 to 255) | (0 to 255) | (0 to 255) |
| DR | Data word (right-hand byte) | (0 to 255) | (0 to 255) | (0 to 255) |
| DW | Data word | (0 to 255) | (0 to 255) | (0 to 255) |
| EF | DB1 parameter: SINEC L1, position of receive mailbox | | | |
| F | Flag - retentive<br>- non-retentive | (0.0 to 63.7)<br>(64.0 to 127.7) | (0.0 to 63.7)<br>(64.0 to 127.7) | (0.0 to 63.7)<br>(64.0 to 255.7) |
| FB | Function block | (0 to 63) | (0 to 63*) | (0 to 255) |
| FB/FY | Flag byte - retentive<br>- non-retentive | (0 to 63)<br>(64 to 127) | (0 to 63)<br>(64 to 127) | (0 to 63)<br>(64 to 255) |
| Formal operand | Expression with a maximum of 4 characters. The first character must be a letter of the alphabet. | | | |
| FW | Flag word - retentive<br>- non-retentive | (0 to 62)<br>(64 to 126) | (0 to 62)<br>(64 to 126) | (0 to 62)<br>(64 to 254) |
| I | Input | (0.0 to 127.7) | (0.0 to 127.7) | (0.0 to 127.7) |
| IB | Input byte | (0 to 127) | (0 to 127) | (0 to 127) |
| IW | Input word | (0 to 126) | (0 to 126) | (0 to 126) |
| KB | Constant (1 byte) | (0 to 255) | (0 to 255) | (0 to 255) |

* +integrated FBs such as CPU 103

| Abbreviation | Explanation | Permissible Operand Value Range for | | |
|---|---|---|---|---|
| | | CPU 100 | CPU 102 | CPU 103 |
| KBE | DB1 parameter: SINEC L1, position of the "Receive" coordination byte | | | |
| KBS | DB1 parameter: SINEC L1, position of the "Send" coordination byte | | | |
| KC | Constant (count) | (0 to 999) | (0 to 999) | (0 to 999) |
| KF | Constant (fixed-point number) | (- 32,768 to +32,767) | (- 32,768 to +32,767) | (- 32,768 to +32,767) |
| KH | Constant (hexadecimal code) | (0 to FFFF) | (0 to FFFF) | (0 to FFFF) |
| KM | Constant (2-byte bit pattern) | (arbitrary bit pattern: 16 bit) | (arbitrary bit pattern: 16 bit) | (arbitrary bit pattern: 16 bit) |
| KS | Constant (2 characters) | (any two alphanumeric characters) | (any two alphanumeric characters) | |
| KT | Constant (time) | (0.0 to 999.3) | (0.0 to 999.3) | (0.0 to 999.3) |
| KY | Constant (2 bytes) | (0 to 255 each byte) | (0 to 255 each byte) | (0 to 255 each byte) |
| LAD | STEP 5-Ladder Diagram | | | |
| NT | DB1 parameter: number of timers being processed | | | |
| OB | Organization block for special applications: 1, 3, 13, 21, 22, 31, 34, 251 | (0 to 63) | (0 to 63) | (0 to 255) |
| OB13 | DB1 parameter: interval (ms) within which OB13 is called and processed | | | |
| OHE | DB1 parameter: enable operating hours counter | | | |
| OHS | DB1 parameter: set operating hours counter | | | |
| OP | Operator panel | | | |
| OV | Overflow. This condition code bit is set if, e.g., a numerical range is exceeded during arithmetic operations. | | | |
| PII | Process image input table | | | |
| PIQ | Process image output table | | | |
| PB | Program block (with block call and return operations) | (0 to 63) | (0 to 63) | (0 to 255) |
| PY | Peripheral byte | | | (0 to 127) |
| PG | Programmer | | | |
| PW | Peripheral word | | | (0 to 126) |
| Q | Output | (0.0 to 127.7) | (0.0 to 127.7) | (0.0 to 127.7) |
| QB | Output Byte | (0 to 127) | (0 to 127) | (0 to 127) |
| QW | Output word | (0 to 126) | (0 to 126) | (0 to 126) |

* +integrated FBs such as CPU 103

| Abbreviation | Explanation | Permissible Operand Value Range for | | |
|---|---|---|---|---|
| | | **CPU 100** | **CPU 102** | **CPU 103** |
| RLO | Result of logic operation | | | |
| RLO affected?<br>Y/N | The RLO is affected/not affected by the operation. | | | |
| RLO dependent?<br>Y<br>Y /Y<br>N | The statement is executed only if the RLO is “1”.<br>The statement is executed only on positive/negative edge change of the RLO.<br>The statement is always executed. | | | |
| RLO reloaded?<br>Y/N | When the next binary operation takes place, the RLO is reloaded/not reloaded (e.g. A I 0.0). | | | |
| RS | System data area<br>- for load operations (supplementary operations) and transfer operations (system operations)<br>- for bit test and set operations (system operations) | | | (0 to 255)<br>(0.0 to 255.15) |
| SAC | STEP address counter | | | |
| SAV | DB1 parameter: Clock time after last switch from STOP RUN or save from last Power OFF | | | |
| SB | Sequence block | | | (0 to 255) |
| SDP | DB1 block ID for system data parameters | | | |
| SET | DB1 parameter: Set clock/date | | | |
| SF | DB1 parameter: SINEC L1, position of send mailbox | | | |
| SL1 | DB1 block ID for SINEC L1 | | | |
| SLN | DB1 parameter: SINEC L1, slave number | | | |
| STL | STEP 5 statement list method of representation | | | |
| STP | DB1 parameter: update the clock while in the STOP state. | | | |
| STW | DB1 parameter: status word location (integral real-time clock) | | | |
| T | Timer<br>- for the “Bit Test” and “Set” supplementary operations | (0 to 15) | (0 to 31) | (0 to 127)<br>(0.0 to 127.15) |
| TFB | DB1 block ID for timer function block | | | |
| TIS | DB1 parameter: set prompt time | | | |
| WD | DB1 parameter: set scan time monitoring | | | |

* +integrated FBs such as CPU 103

## B Dimension Drawings

**Figures**


| | | |
|---|---|---|
| B-1 | Cross Sections of Standard Mounting Rails | B - 1 |
| B-2 | Dimension Drawing of the 483-mm (19-in.) Standard Mounting Rail | B - 1 |
| B-3 | Dimension Drawing of the 530-mm (20.9-in.) Standard Mounting Rail | B - 2 |
| B-4 | Dimension Drawing of the 830-mm (32.7-in.) Standard Mounting Rail | B - 2 |
| B-5 | Dimension Drawing of the 2-m (6.6-ft.) Standard Mounting Rail | B - 2 |
| B-6 | Dimension Drawing of the S5-100U (CPU) | B - 3 |
| B-7 | Dimension Drawing of the Bus Unit (Crimp Snap-in Connections) with I/O Module | B - 4 |
| B-8 | Dimension Drawing of the Bus Unit (SIGUT Screw-type Terminals) with I/O Module | B - 5 |
| B-9 | Dimension Drawing of the IM 315 Interface Module | B - 6 |
| B-10 | Dimension Drawing of the IM 316 Interface Module (6ES5 316-8MA12) | B - 7 |
| B-11 | Dimension Drawing of the PS 930 and PS 931 Power Supply Modules | B - 8 |

# B Dimension Drawings

Dimensions are indicated in millimeters. The approximate equivalent in inches is indicated in parentheses. (1 mm=0.039 in. rounded off to the nearest tenth or hundredth of an inch)

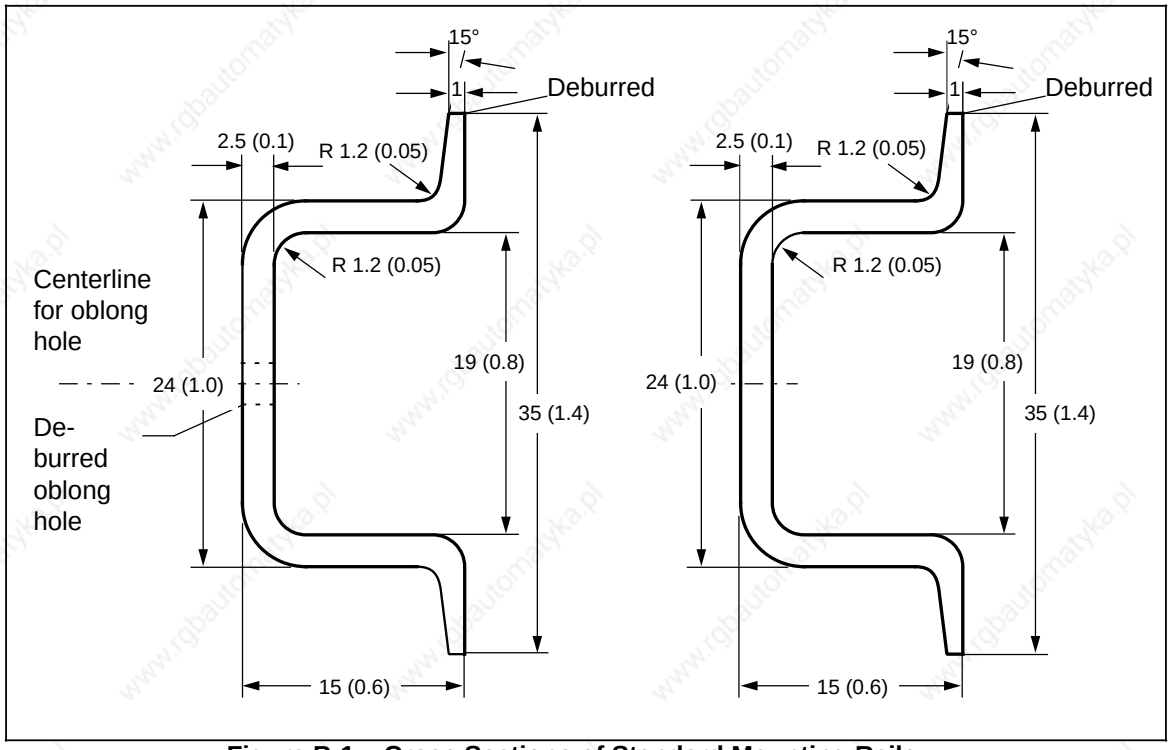



**Figure B-1. Cross Sections of Standard Mounting Rails**

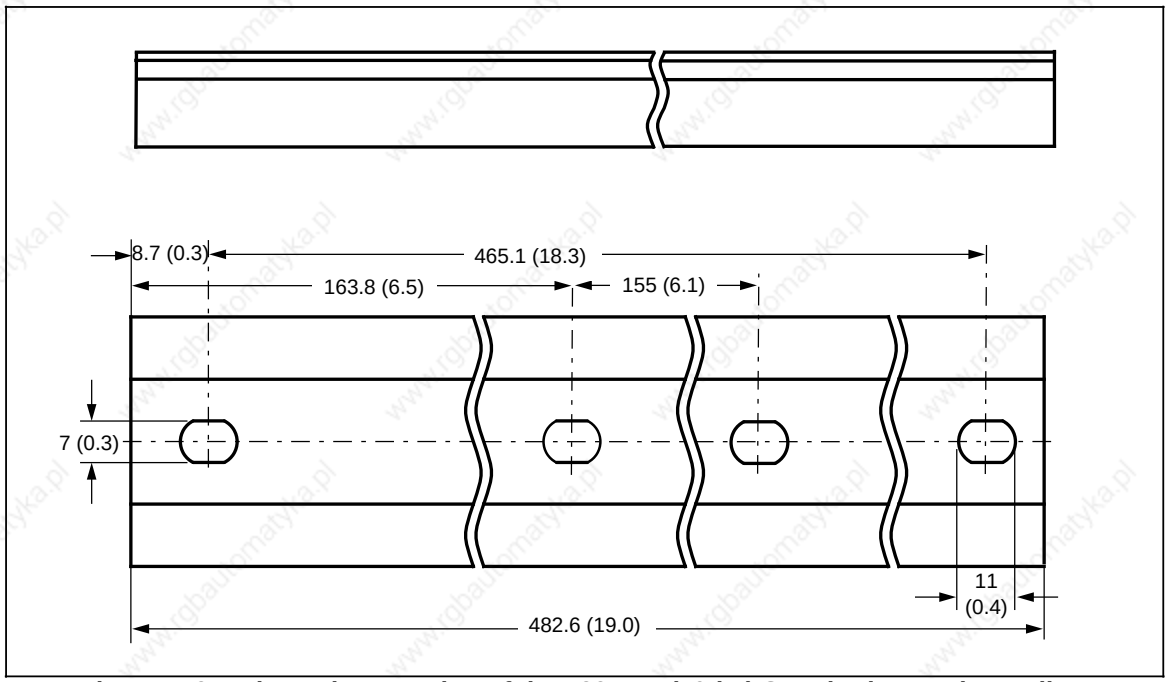



**Figure B-2. Dimension Drawing of the 483-mm (19-in.) Standard Mounting Rail**

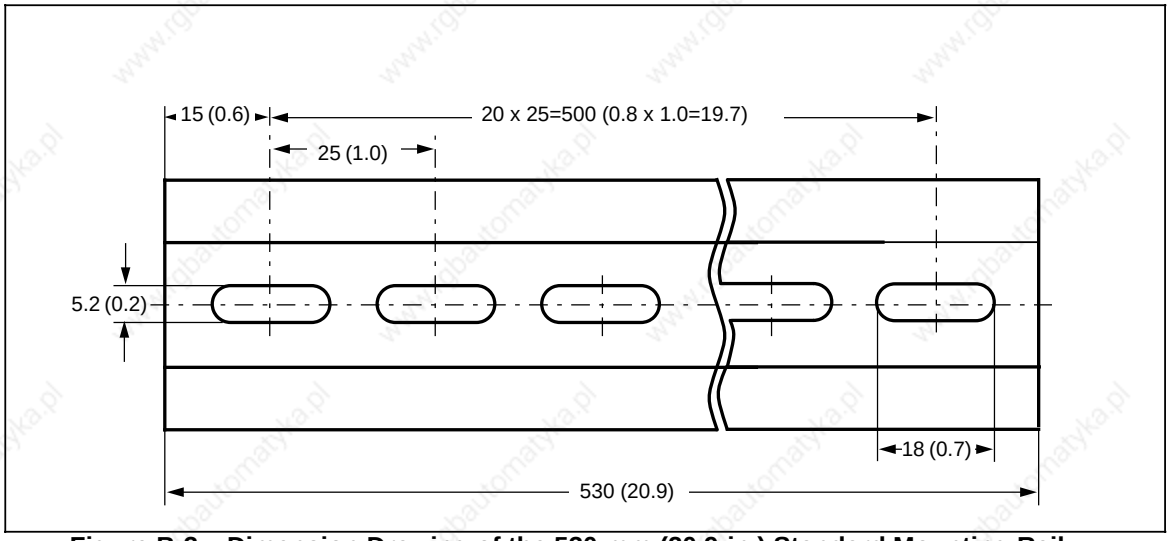



**Figure B-3. Dimension Drawing of the 530-mm (20.9-in.) Standard Mounting Rail**

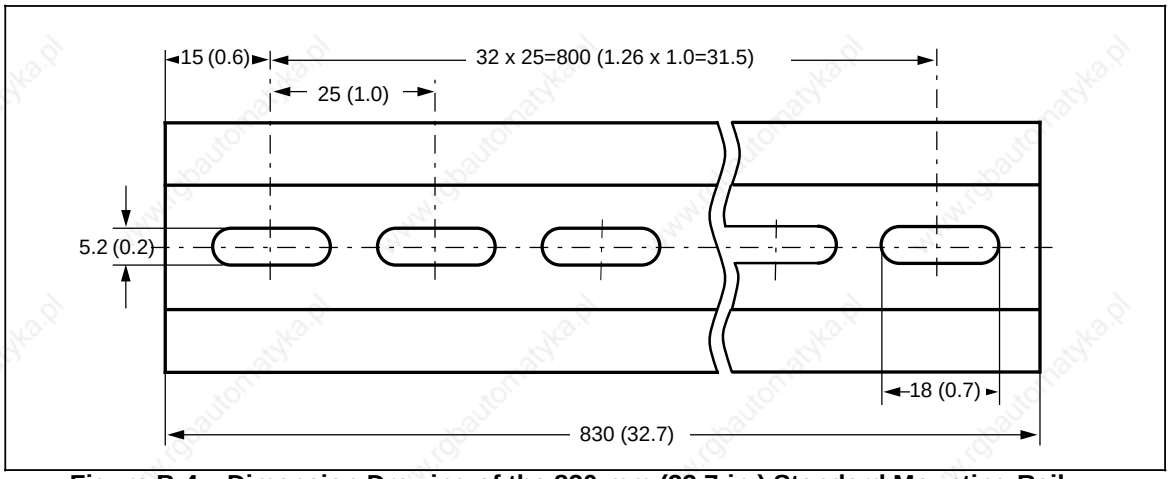



**Figure B-4. Dimension Drawing of the 830-mm (32.7-in.) Standard Mounting Rail**

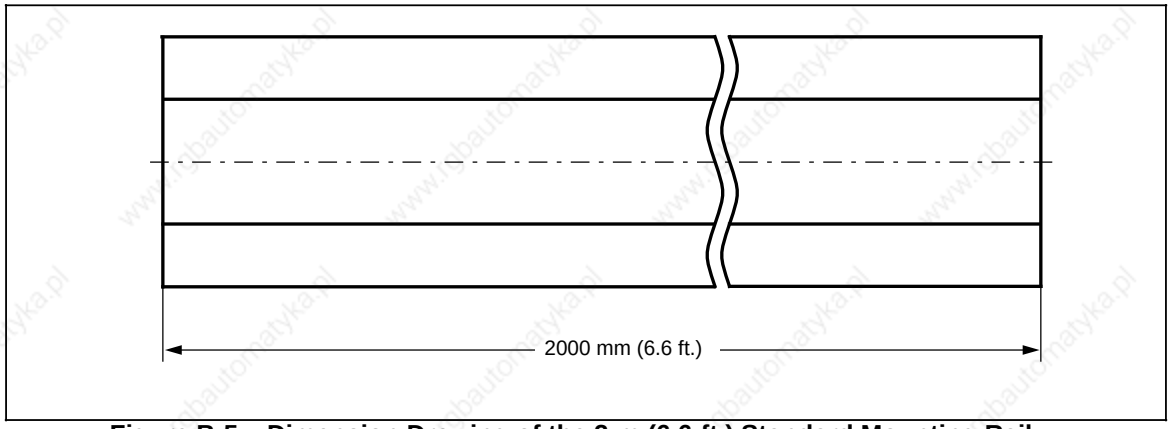



**Figure B-5. Dimension Drawing of the 2-m (6.6-ft.) Standard Mounting Rail**

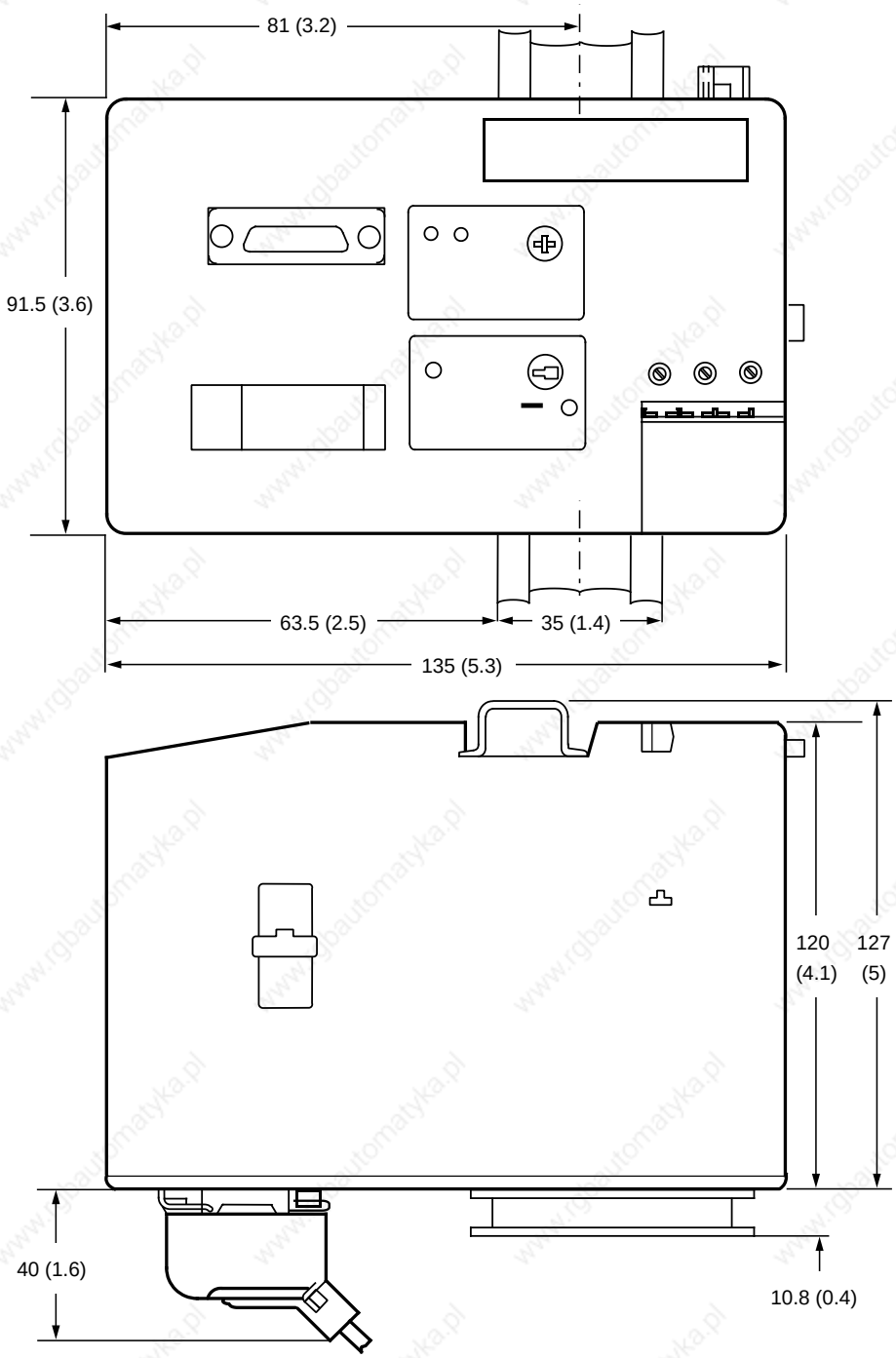



**Figure B-6. Dimension Drawing of the S5 100U (CPU)**

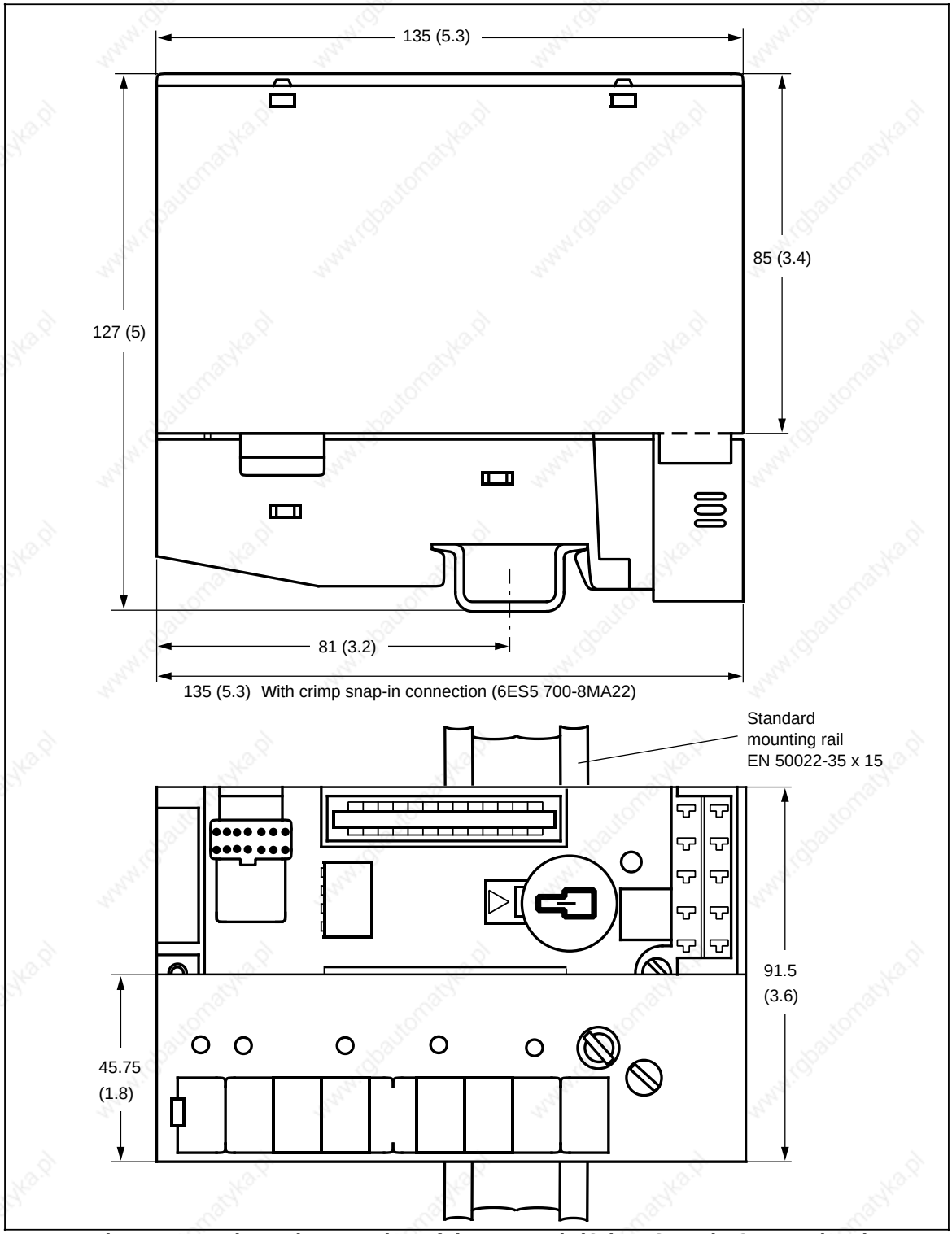



**Figure B-7. Dimension Drawing of the Bus Unit (Crimp Snap-in Connections) with I/O Module**

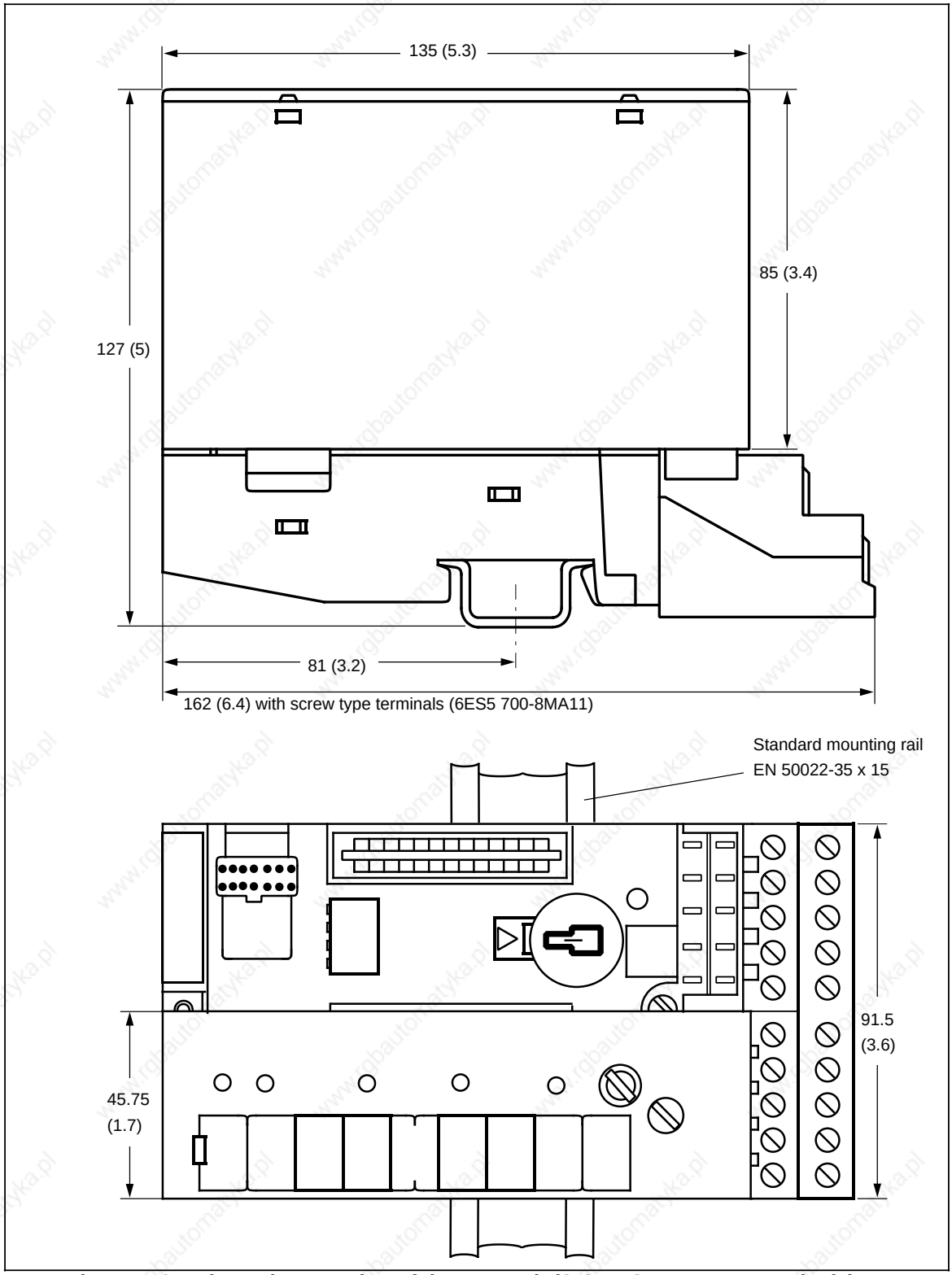



**Figure B-8. Dimension Drawing of the Bus Unit (SIGUT Screw-type Terminals) with I/O Module**

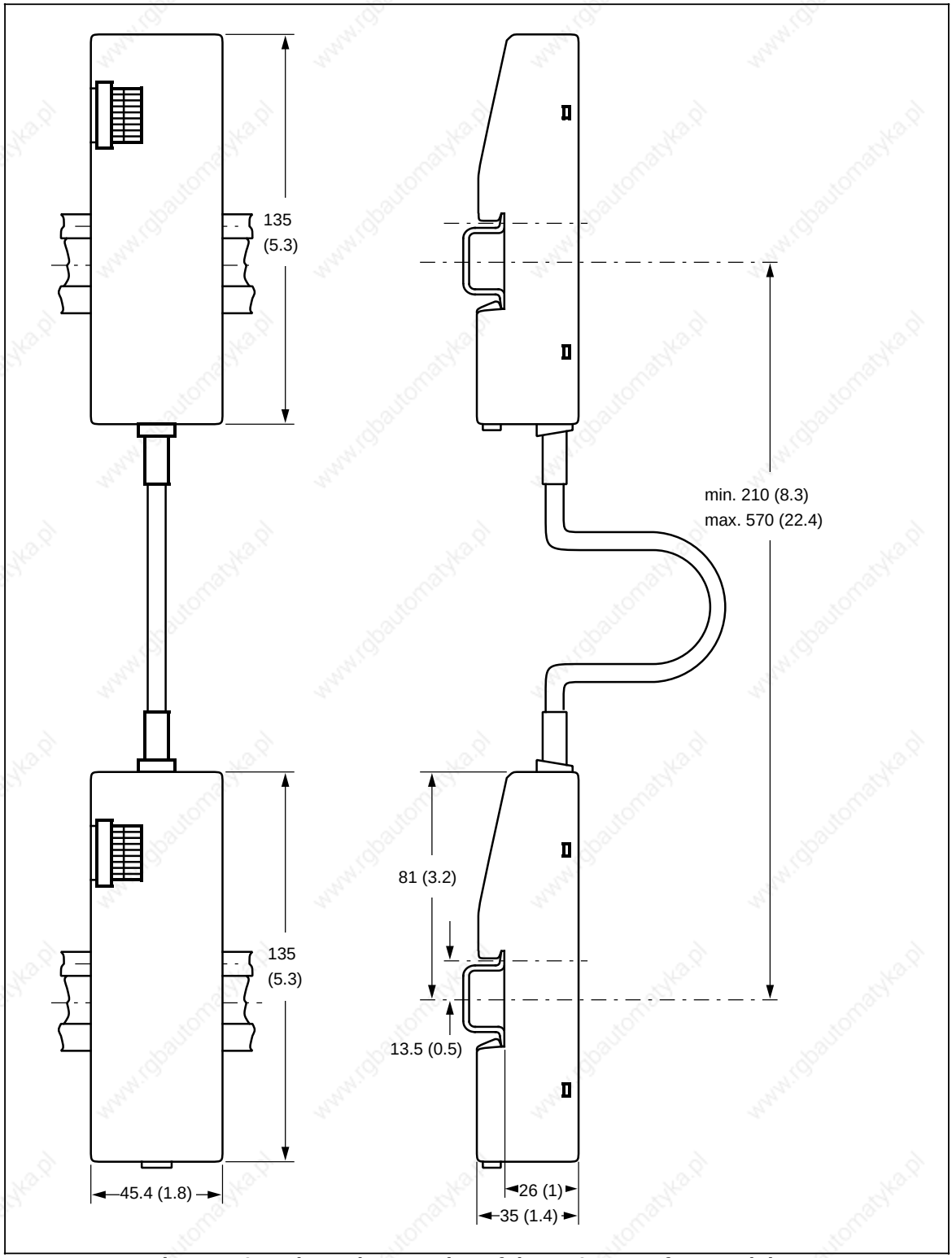



**Figure B-9. Dimension Drawing of the IM 315 Interface Module**

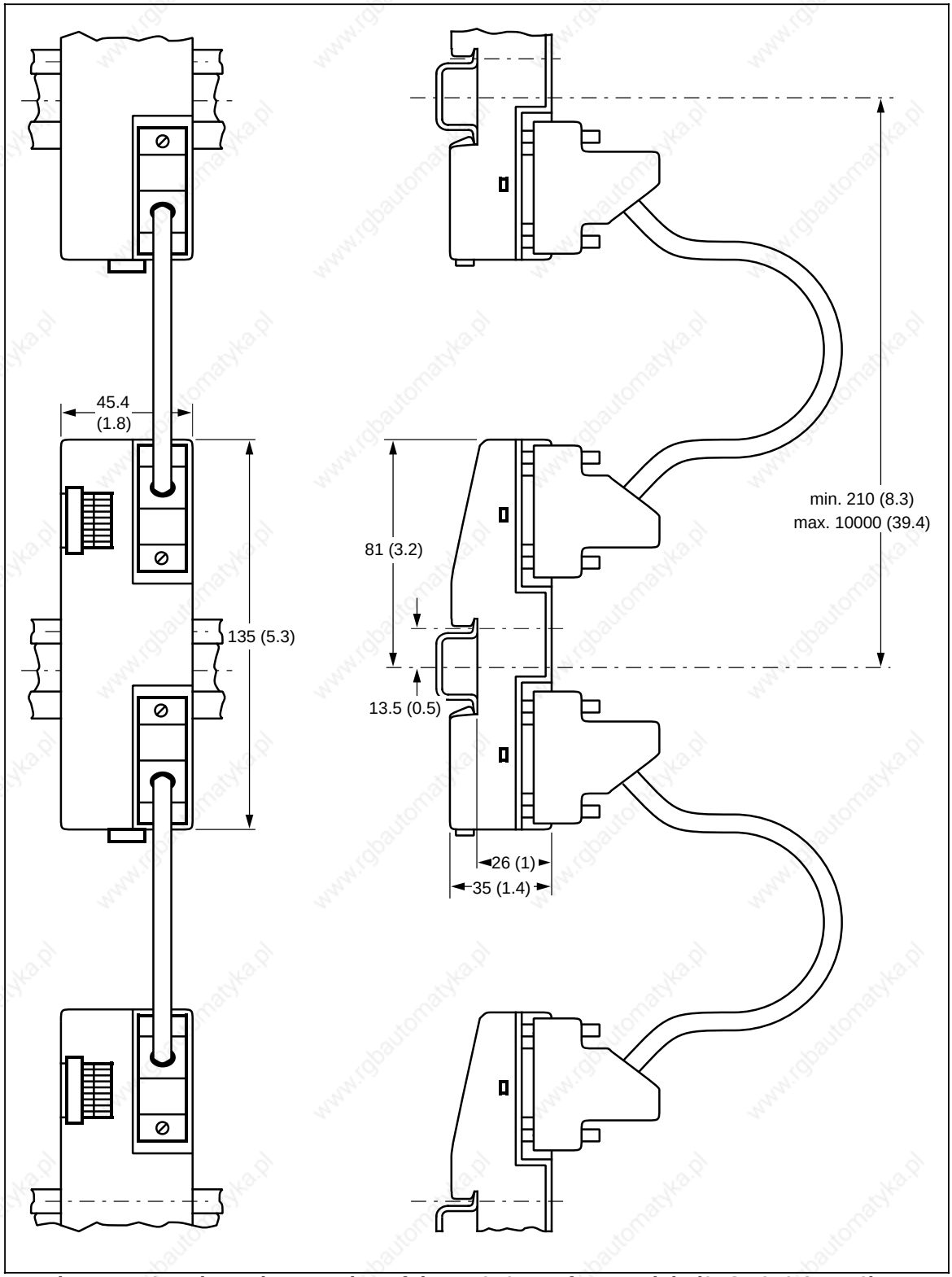



**Figure B-10. Dimension Drawing of the IM 316 Interface Module (6ES5 316-8MA12)**

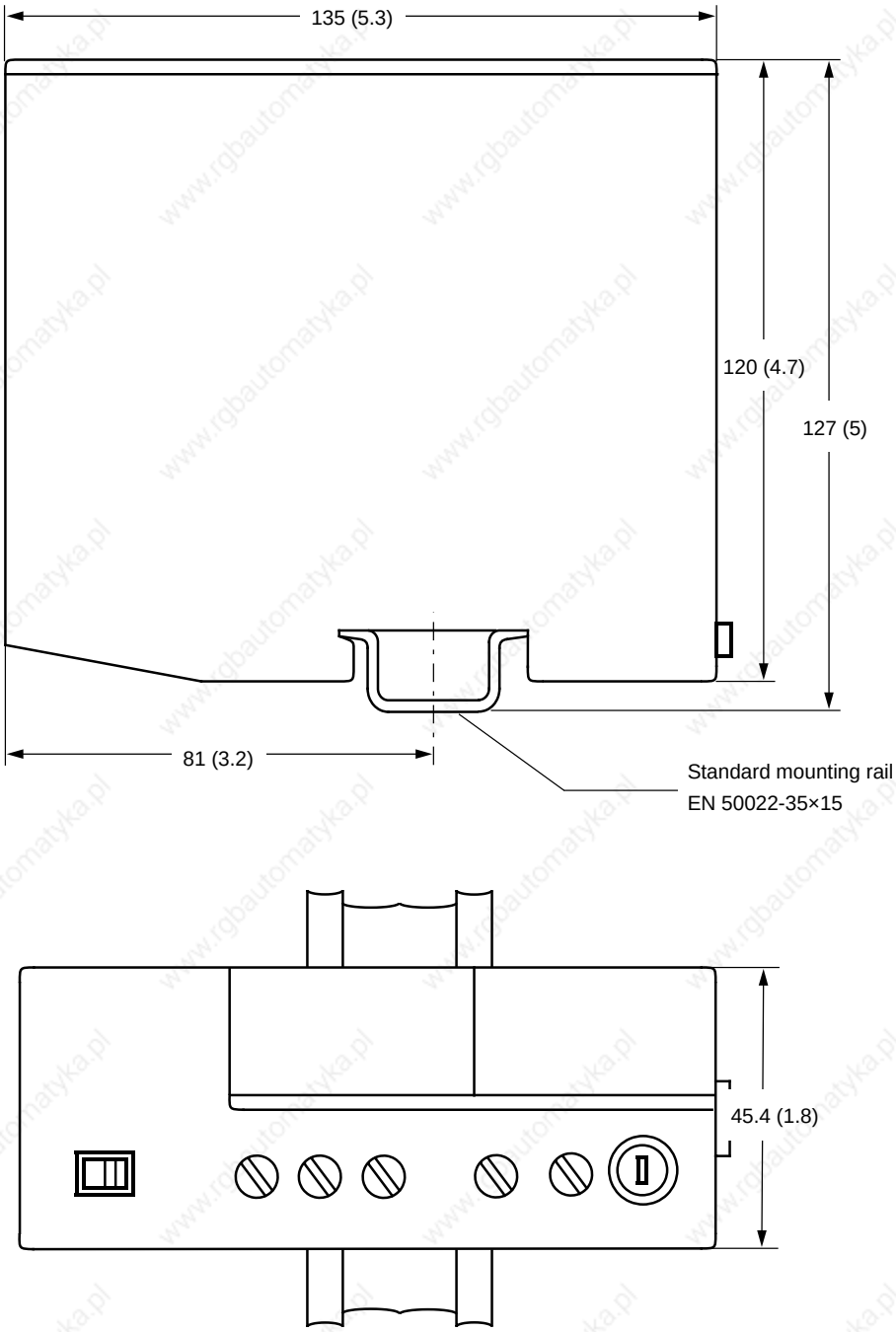



**Figure B-11. Dimension Drawing of the PS 930 and PS 931 Power Supply Modules**

# C Active and Passive Faults in Automation Equipment

# C Active and Passive Faults in Automation Equipment / Guidelines for Handling Electrostatic Sensitive Devices

## Active and Passive Faults in Automation Equipment

- Depending on the particular task for which the electronic automation equipment is used, both **active** as well as **passive** faults can result in a **dangerous** situation. For example, in drive control, an active fault is generally dangerous because it can result in an unauthorized startup of the drive. On the other hand, a passive fault in a signalling function can result in a dangerous operating state not being reported to the operator.

- The differentiation of the possible faults and their classification into dangerous and non-dangerous faults, depending on the particular task, is important for all safety considerations in respect to the product supplied.

**Warning**

In all cases where a fault in automation equipment can result in severe personal injury or substantial property damage, i.e., where a dangerous fault can occur, additional external measures, additional external measures must be taken or equipment provided to ensure or force safe operating conditions even in the event of a fault (e.g., by means of independent limit monitors, mechanical interlocks, etc.).

**Procedures for Maintenance and Repair**

If you are carrying out measurement or testing work on an active unit, you must adhere to the rules and regulations contained in the “VGB 4.0 Accident Prevention Regulations“ of the German employers liability assurance association (“Berufsgenossenschaften“). Pay particular attention to paragraph 8, “Permissible exceptions when working on live parts.“

Do not attempt to repair an item of automation equipment. Such repairs may only be carried out by **Siemens service personnel or repair shops Siemens has authorized to carry out such repairs.**

The information in this manual is checked regularly for updating and correctness and may be modified without prior notice. The information contained in this manual is protected by copyright. Photocopying and translation into other languages is not permitted without express permission from Siemens.

# Guidelines for Handling Electrostatic Sensitive Devices (ESD)

**What is ESD?**

All electronic modules are equipped with large-scale integrated ICs or components. Due to their design, these electronic elements are very sensitive to overvoltages and thus to any electrostatic discharge.

These **E**lectrostatic **S**ensitive **D**evices are commonly referred to by the abbreviation **ESD**.

Electrostatic sensitive devices are labelled with the following symbol:

**Caution**

Electrostatic sensitive devices are subject to voltages that are far below the voltage values that can still be perceived by human beings. These voltages are present if you touch a component module without previously being electrostatically discharged. In most cases, the damage caused by an overvoltage is not immediately noticeable and results in total damage only after a prolonged period of operation.

**Electrostatic charging of objects and persons**

Every object with no conductive connection to the electrical potential of its surroundings can be charged electrostatically. In this way, voltages up to 15000 V can build up whereas minor charges, i.e. up to 100 V, are not relevant.

**Examples:**

| | |
|---|---|
| • Plastic covers | up to 5000 V |
| • Plastic cups | up to 5000 V |
| • Plastic-bound books and notebooks | up to 8000 V |
| • Desoldering device with plastic parts | up to 8000 V |
| • Walking on plastic flooring | up to 12000 V |
| • Sitting on a padded chair | up to 15000 V |
| • Walking on a carpet (synthetic) | up to 15000 V |

**Limits for perceiving electrostatic discharges**

An electrostatic discharge is

- perceptible from 3500 V
- audible from 4500 V
- visible from 5000 V

A fraction of those voltages is capable of destroying or damaging electronic devices.

Carefully note ad apply the protective measures described below to protect and prolong the life of your modules and components.

**General protective measures against electrostatic discharge damage**

- Keep plastics away from sensitive devices. Most plastic materials have a tendency to build up electrostatic charges easily.

- Make sure that the personnel, working surfaces and packaging are sufficiently grounded when handling electrostatic sensitive devices.

- If possible, avoid any contact with electrostatic sensitive devices. Hold modules without touching the pins of components or printed conductors. In this way, the discharged energy cannot affect the sensitive devices.

**Additional precautions for modules without housings**

Note the following measures that have to be taken for modules that are not protected against accidental contact:

- Touch electrostatical sensitive devices only
  - if you wear a wristband complying with ESD specifications or
  - if you use special ESD footwear or ground straps when walking ona an ESD floor.

- Persons working on electronic devices should first discharge their bodies by touching grounded metallic parts (e.g. bare metal parts of switchgear cabinets, water pipes, etc.)

- Protect the modules against contact with chargeable and highly insulating materials, such as plastic foils, insulating table tops or clothes made of plastic fibres.

- Place electrostatic sensitive devices only on conductive surfaces:
  - Tables with ESD surfaces
  - Conductive ESD foam plastic (ESD foam plastic ismostly coloured black)
  - ESD bags

- Avoid direct contact of eletrostatic sensitive devices with visual display units, monitors or TV sets (minimum distance to screen > 10 cm).

The following Figures once again illustrates the precautions for handling electrostatically sensitive devices.

a Conductive flooring material
b Table with conductive, grounded surface
c ESD footwear
d ESD smock
e Gounded ESD writstband
f Grounded connection of switchgear cabinet
g Grounded chair

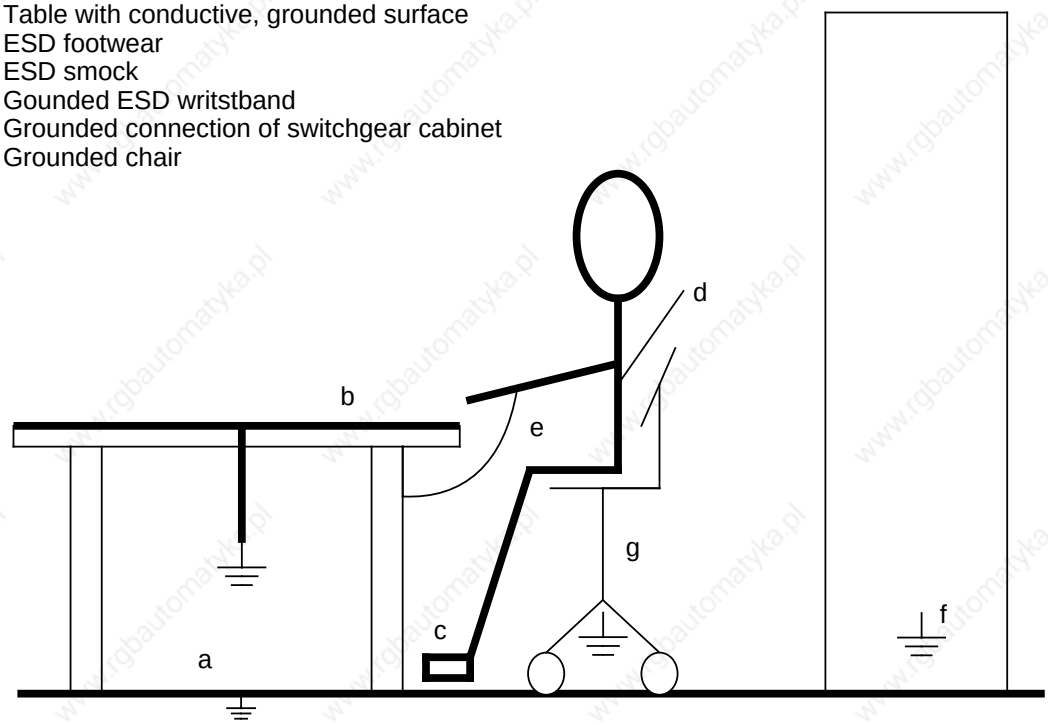



**Figure C-1. ESD Measures**

**Taking measurements and working on ESD modules**

Measurements may be taken on electrostatic sensitive devices only if

- the measuring device is grounded (e.g. via protective conductor) or
- the tip of the isolated measuring tool has previously been discharged (e.g. by briefly touching grounded metal parts).

## D Information for Ordering Accessories

# D Information for Ordering Accessories

| | Order Numbers |
|---|---|
| **Standard 35 mm Mounting Rail** | |
| for 19-in. cabinets, length 483 mm | 6ES5 710-8MA11 |
| for 600 mm cabinets, length 530 mm | 6ES5 710-8MA21 |
| for 900 mm cabinets, length 830 mm | 6ES5 710-8MA31 |
| Length 2000 mm, without holes | 6ES5 710-8MA41 |
| **Power Supply Modules** | |
| Power supply module PS 930 | |
| 115/230 V AC; 1 A | 6ES5 930-8MD11 |
| Replacement fuse (3A extra-fast) | 6ES5 980-3BC61 |
| Power supply module PS 931 | |
| 115/230 V AC; 24 V DC; 2 A | 6ES5 931-8MD11 |
| (with electronic circuit protection) | |
| Power supply module PS 935 | |
| 24 V DC; 9 V DC, 2.5 A | 6ES5 935-8ME11 |
| Load power supply 6EW1 | |
| 115/230 V AC; 24 V DC; 4 A | 6EW1 380-1AB |
| 115/230 V AC; 24 V DC; 10 A | 6EW1 380-4AB01 |
| **Bus Units** | |
| Bus unit with SIGUT screw-type terminals | 6ES5 700-8MA11 |
| Bus unit with crimp snap-in terminals | 6ES5 700-8MA22 |
| Interrupt bus unit, with SIGUT screw-type terminals | 6ES5 700-8MB11 |
| Interrupt bus unit with crimp snap-in terminals | 6ES5 700-8MB21 |
| **Accessories** | |
| Extracting tool | |
| for crimp snap-in connections | 6ES5 497-8MA11 |
| Crimp snap-in contacts, 250 pieces | 6XX3 070 |
| Crimping tool | |
| for attaching the crimp contacts | 6XX3 071 |
| **Interface Modules** | |
| IM 315 interface module | 6ES5 315-8MA11 |
| IM 316 interface module | 6ES5 316-8MA12 |
| - Cable connectors (0.5 m/1.6 ft.) | 6ES5 712-8AF00 |
| - Cable connectors (2.5 m/8.2 ft.) | 6ES5 712-8BC50 |
| - Cable connectors (5.0 m/16.5 ft.) | 6ES5 712-8BF00 |
| - Cable connectors (10 m/33 ft.) | 6ES5 712-8CB00 |

| | | Order Numbers |
|---|---|---|
| **Central Processing Units (CPUs)** | | |
| **CPU 100** | | 6ES5 100-8MA02 |
| **CPU 102** | | 6ES5 102-8MA02 |
| **CPU 103** | | 6ES5 103-8MA03 |
| **S5-100U System Manual (CPU 100, CPU 102, CPU 103)** | | |
| German | | 6ES5 998-0UB13 |
| English | | 6ES5 998-0UB23 |
| French | | 6ES5 998-0UB33 |
| Spanish | | 6ES5 998-0UB43 |
| Italian | | 6ES5 998-0UB53 |
| **Accessories for the CPUs** | | |
| Back-up battery lithium AA; 3.4 V/850 mAh | | 6ES5 980-0MB11 |
| Memory submodule (EPROM) | 4096 statements | 6ES5 375-1LA15 |
| Memory submodule (EPROM) | 8192 statements | 6ES5 375-1LA21 |
| Memory submodule (EPROM) | 16384 statements | 6ES5 375-1LA41 |
| Memory submodule (EEPROM) | 1024 statements | 6ES5 375-0LC11 |
| Memory submodule (EEPROM) | 2048 statements | 6ES5 375-0LC21 |
| Memory submodule (EEPROM) | 4096 statements | 6ES5 375-0LC31 |
| Memory submodule (EEPROM) | 8192 statements | 6ES5 375-0LC41 |

| | | Order Numbers |
|---|---|---|
| **Manual for IP 262 Closed-Loop Control Module** | | |
| | German | 6ES5 998-5SG11 |
| | English | 6ES5 998-5SG21 |
| | French | 6ES5 998-5SG31 |
| | Italian | 6ES5 998-5SG51 |
| **Manual for IP 263 Positioning Module** | | |
| | German | 6ES5 998-5SK11 |
| | English | 6ES5 998-5SK21 |
| **Manual for IP 264 Electronic Cam Controller Module** | | |
| | German | 6ES5 998-5SL11 |
| | English | 6ES5 998-5SL21 |
| **Manual for IP 265 Closed-Loop Control Module** | | |
| | German | 6ES5 998-5SH11 |
| | English | 6ES5 998-5SH21 |
| | French | 6ES5 998-5SH31 |
| **Manual for IP 266 Positioning Module** | | |
| | German | 6ES5 998-5SC11 |
| | English | 6ES5 998-5SC21 |
| **Manual for IP 267 Stepper Motor Module** | | |
| | German | 6ES5 998-5SD11 |
| | English | 6ES5 998-5SD21 |
| | French | 6ES5 998-5SD31 |
| | Spanish | 6ES5 998-5SD41 |
| **Digital Input Modules** | | |
| 8 x 5 to 24 V DC | isolated | 6ES5 433-8MA11 |
| 4 x 24 V DC | | 6ES5 420-8MA11 |
| 8 x 24 V DC | | 6ES5 421-8MA12 |
| 16 x 24 V DC | | 6ES5 422-8MA11 |
| 8 x 24 V DC | isolated | 6ES5 431-8MA11 |
| 4 x 24 to 60 V DC | isolated | 6ES5 430-8MB11 |
| 4 x 115 V AC | isolated | 6ES5 430-8MC11 |
| 8 x 115 V AC | isolated | 6ES5 431-8MC11 |
| 4 x 230 V AC | isolated | 6ES5 430-8MD11 |
| 8 x 230 V AC | isolated | 6ES5 431-8MD11 |
| **Digital Output Modules** | | |
| 8 x 5 to 24 V DC/0.1 A | isolated | 6ES5 453-8MA11 |
| 4 x 24 V DC/0.5 A | | 6ES5 440-8MA12 |
| 4 x 24 V DC/2 A | | 6ES5 440-8MA22 |
| 8 x 24 V DC/0.5 A | | 6ES5 441-8MA11 |
| 8 x 24 V DC/0.5 A | isolated | 6ES5 451-8MA11 |
| 4 x 24 to 60 V DC/0.5 A | isolated | 6ES5 450-8MB11 |
| 4 x 115 to 230 V AC/1 A | isolated* | 6ES5 450-8MD11 |
| 8 x 115 to 230 V AC/0.5 A | isolated* | 6ES5 451-8MD11 |
| 4 relays x 30 V DC/230 V AC | | 6ES5 452-8MR11 |
| 8 relays x 30 V DC/230 V AC | | 6ES5 451-8MR12 |
| * Replacement fuse (10 A extra-fast) | | 6ES5 980-3BC11 |

| | | Order Numbers |
|---|---|---|
| **Digital Input/Output Module** | | |
| 24 V DC 16 inputs/16 outputs | | 6ES5 482-8MA13 |
| **Accessories** | | |
| Front connector, 40-pin for crimp snap-in connection | | |
| – with crimp contacts | | 6ES5 490-8MA13 |
| – without crimp contacts | | 6ES5 490-8MA03 |
| Front connector, 40-pin for screw-type connection | | |
| – standard | | 6ES5 490-8MB11 |
| – increased EMC strength | | 6ES5 490-8FB11 |
| **Analog Input Modules** | | |
| 4 x ± 50 mV | isolated | 6ES5 464-8MA11 |
| 4 x ± 50 mV | isolated | 6ES5 464-8MA21 |
| 4 x ± 1 V | isolated | 6ES5 464-8MB11 |
| 4 x ± 10 V | isolated | 6ES5 464-8MC11 |
| 4 x ± 20 mA | isolated | 6ES5 464-8MD11 |
| 4 x + 4 to 20 mA | isolated | 6ES5 464-8ME11 |
| 2 x PT 100/± 500 mV | isolated | 6ES5 464-8MF11 |
| 2 x PT 100/± 500 mV | isolated | 6ES5 466-8MF21 |
| 4 x + 0 to 10 V | isolated | 6ES5 466-8MC11 |
| **Analog Output Modules** | | |
| 2 x ± 10 V | isolated | 6ES5 470-8MA12 |
| 2 x ± 20 mA | isolated | 6ES5 470-8MB12 |
| 2 x + 4 to 20 mA | isolated | 6ES5 470-8MC12 |
| 2 x + 1 to 5 V | isolated | 6ES5 470-8MD12 |
| **Function Modules** | | |
| IP 262 Closed-loop control module | | |
| – with 3 analog outputs | | 6ES5 262-8MA12 |
| – with 8 binary outputs | | 6ES5 262-8MB12 |
| IP 263 Positioning module | | 6ES5 263-8MA13 |
| IP 264 Electronic cam controller module | | 6ES5 264-8MA12 |
| IP 265 High Speed Sub Control | | 6ES5 265-8MA01 |
| IP 266 Positioning module | | 6ES5 266-8MA11 |
| IP 267 Stepper motor control module | | 6ES5 267-8MA11 |
| Diagnostic module 330 | | 6ES5 330-8MA11 |
| Timer module 380 2 x 0.3 to 300 s | | 6ES5 380-8MA11 |
| Counter module 2 x 0 to 500 Hz | | 6ES5 385-8MA11 |
| Counter module 385B 1 x 25/500 KHz | | 6ES5 385-8MB11 |
| Comparator module 461 2 x 1 to 20 mA/0.5 to 10 V | | 6ES5 461-8MA11 |
| CP 521 SI Printer output module | | 6ES5 521-8MA22 |
| CP 521 BASIC communications module | | 6ES5 521-8MB12 |
| Simulator 788 (digital input/output signals) | | 6ES5 788-8MA11 |

| | Order Numbers |
|---|---|
| **Operator Panels and Programmers** | |
| OP 393-III Operator Panel | |
| with connecting cable | 6ES5 393-0UA15 |
| OP 393-III Operator guide, German | 6ES5 998-0UQ12 |
| PG 605U Programmer | 6ES5 605-0UA11 |
| PG 605U Operator guide, German | 6ES5 998-0UP11 |
| PG 720 Programmer | 6ES7 720-0AB00-0YA0 |
| PG 720 C Programmer | 6ES7 720-1AB00-0YB0 |
| PG 740 Programmer | 6ES7 740-0AA00-0YA0 |
| PG 760 Programmer | 6ES7 760-1AA00-0YA0 |
| (–> Catalog ST 59) | |
| Cable connector for connecting the OP 393-III to the CPU | |
| 3 m | 6ES5 728-0BD00 |
| 10 m | 6ES5 728-0CB00 |
| 20 m | 6ES5 728-0CC00 |
| Special lengths up to 1000 m (–> Catalog ST 80) | |
| **Program Packages** | |
| **Basic Functions Program Package** | |
| with description in German, English, and French | |
| for the S5-DOS operating system | 6ES5 848-8AA01 |
| for the MS-DOS, S5-DOS/MT operating system | 6ES5 848-7AA01 |
| **Floating Point Arithmetic Program Package** | |
| with description in German, English, and French | |
| for the S5-DOS operating system | 6ES5 845-8GP01 |
| for the MS-DOS, S5-DOS/MT operating system | 6ES5 845-7GP01 |
| **GRAPH 5 Program Package** | |
| with description in German, English, and French | 6ES5 886-1FA01 |
| **GRAPH 5 Mini Program Package** | |
| German | 6ES5 886-1SE11 |
| English | 6ES5 886-1SE21 |
| French | 6ES5 886-1SE31 |
| Spanish | 6ES5 886-1SE41 |
| Italian | 6ES5 886-1SE51 |
| **S5-100U Program Package** | |
| with description in | |
| German | 6ES5 840-4BC11 |
| English | 6ES5 840-4BC21 |
| Italian | 6ES5 840-4BC51 |
| **STEP 5 Package for Mini PLCs for PC** | 6ES5 866-0MA02 |
| Documentation for STEP 5 Package, German | 6ES5 896-0MY11 |

# E Reference Materials

# E Reference Materials

**The following reference material can be ordered from your local Siemens Company or your local bookshop:**

- **Automating with the SIMATIC® S5-115U**
  Programmable Controllers
  Hans Berger
  Siemens AG, Berlin and Munich, 1989 (2nd Edition)
  (Order No.: ISBN 3-8009-1530-8)

- **Programmable Controllers**
  Basic Concepts
  Siemens AG, 1992 (Order No.: A19100-L531-F914-X-7600)

- **Programming Primer for the SIMATIC® S5-90/95U**
  Practical Exercises with the PG 710 Programmer
  Siemens AG, Berlin and Munich, 1992
  Order No.: A19100-L531-F550-X-7600

## F Siemens Addresses Worldwide

# F Siemens Addresses Worldwide

## European Companies and Representatives

**Austria**
Siemens AG Österreich
**Vienna**
**Bregenz**
**Graz**
**Innsbruck**
**Klagenfurt**
**Linz**
**Salzburg**

**Belgium**
Siemens S.A.
**Brussels**
**Liège**
Siemens N.V.
**Brussels**
**Antwerp**
**Gent**

**Bulgaria**
RUEN office of the INTERPRED corporation, agency of the Siemens AG Sofia
**Sofia**

**Czechoslovakia**
EFEKTIM Engineering Consultants, Siemens AG
**Prague**

**Denmark**
Siemens A/S
**Copenhagen,** Ballerup
**Hojbjerg**

**Federal Republic of Germany**
Branch offices of the Siemens AG
**Berlin**
**Bremen**
**Cologne**
**Dortmund**
**Düsseldorf**
**Essen**
**Frankfurt/Main**
**Hamburg**

**Federal Republic of Germany** (continued)
**Hanover**
**Leipzig**
**Mannheim**
**Munich**
**Nuremberg**
**Saarbrücken**
**Stuttgart**

**Finland**
Siemens Osakeyhtiö
**Helsinki**

**France**
Siemens S.A.
**Paris,** Saint-Denis
**Lyon,** Caluire-et-Cuire
**Marseilles**
**Metz**
**Seclin** (Lille)
**Strasbourg**

**Great Britain**
Siemens Ltd.
**London,** Sunbury-on-Thames
**Birmingham**
**Bristol,** Clevedon
**Congleton**
**Edinburgh**
**Glasgow**
**Leeds**
**Liverpool**
**Newcastle**

**Greece**
Siemens A.E.
**Athens**
**Thessaloniki**

**Hungary**
SICONTACT GmbH
**Budapest**

**Iceland**
Smith & Norland H/F
**Reykjavik**

**Ireland**
Siemens Ltd.
**Dublin**

**Italy**
Siemens S. p. A.
**Milan**
**Bari**
**Bologna**
**Brescia**
**Casoria**
**Florence**
**Genoa**
**Macomer**
**Padua**
**Rome**
**Turin**

**Luxemburg**
Siemens S.A.
**Luxembourg**

**Malta**
J.R. Darmanin & Co., Ltd.
**Valletta**

**Netherlands**
Siemens Nederland N.V.
**The Hague**

**Norway**
Siemens A/S
**Oslo**
**Bergen**
**Stavanger**
**Trondheim**

**Poland**
PHZ Transactor S.A.
**Warsaw**
**Gda sk-Letnica**
**Katowice**

**Portugal**
Siemens S.R.A.L.
**Lisbon**
**Faro**
**Leiria**
**Porto**

**Romania**
Siemens birou de consulta ii tehnice
**Bukarest**

**Spain**
Siemens S.A.
**Madrid**

**Sweden**
Siemens AB
**Stockholm**
**Eskilstuna**
**Göteborg**
**Jönköping**
**Luleå**
**Malmö**
**Sundsvall**

**Switzerland**
Siemens-Albis AG
**Zürich**
**Bern**
Siemens-Albis S.A.
**Lausanne,** Renens

**Turkey**
ETMA
**Istanbul**
**Adana**
**Ankara**
**Bursa**
**Izmir**
**Samsun**

**USSR**
Siemens AG Agency
**Moscow**

**Yugoslavia**
General Export
OOUR Zastupstvo
**Belgrade**
**Ljubljana**
**Rijeka**
**Sarajewo**
**Skopje**
**Zagreb**

## Non-European Companies and Representatives

### Africa

**Algeria**
Siemens Bureau
Alger
**Algier**

**Angola**
Tecnidata
**Luanda**

**Burundi**
SOGECOM
**Bujumbara**

**Egypt**
Siemens Resident Engineers
**Cairo-Mohandessin**
**Alexandria**
Centech
**Zamalek-Cairo**

**Ethiopia**
Addis Electrical Engineering Ltd.
**Addis Abeba**

**Ivory Coast**
Siemens AG
Succursale Côte d'Ivoire
**Abidjan**

**Kenya**
Achelis (Kenya) Ltd.
**Nairobi**

**Libya**
Siemens AG
Branch Office Libya
**Tripoli**

**Mauritius**
Rey & Lenferna Ltd.
**Port Louis**

**Morocco**
SETEL
Société Electrotechnique et de Télécommunications S.A.
**Casablanca**

**Mozambique**
Siemens Resident Engineer
**Maputo**

**Namibia**
Siemens Resident Engineer
**Windhoek**

**Nigeria**
Electro Technologies Nigeria Ltd. (Eltec)
**Lagos**

**Rwanda**
Etablissement Rwandais
**Kigali**

**Simbabwe**
Electro Technologies Corporation (Pvt.) Ltd.
**Harare**

**South Africa**
Siemens Ltd.
**Johannesburg**
**Cape Town**
**Durban**
**Middleburg**
**Newcastle**
**Port Elizabeth**
**Pretoria**

**Sudan**
National Electrical & Commercial Company (NECC)
**Khartoum**

**Swaziland**
Siemens (Pty.) Ltd.
**Mbabane**

**Tanzania**
Tanzania Electrical Services Ltd.
**Dar-es-Salaam**

**Tunesia**
Sitelec S.A.
**Tunis**

**Zaire**
SOFAMATEL S.P.R.L.
**Kinshasa**

**Zambia**
Electrical Maintenance Lusaka Ltd.
**Lusaka**
Mining projects:
General Mining Industries Ltd.
**Kitwe**

## America

**Argentina**
Siemens S.A.
**Buenos Aires**
**Bahía Blanca**
**Córdoba**
**Mendoza**
**Rosario**

**Bolivia**
Sociedad Comercial e Industrial Hansa Ltd.
**La Paz**

**Brazil**
Siemens S.A.
**São Paulo**
**Belém**
**Belo Horizonte**
**Brasília**
**Campinas**
**Curitiba**
**Florianópolis**
**Fortaleza**
**Porto Alegre**
**Recife**
**Rio de Janeiro**
**Salvador de Bahía**
**Vitoria**

**Canada**
Siemens Electric Ltd.
**Montreal,** Québec
**Toronto,** Ontario

**Chile**
INGELSAC
**Santiago de Chile**

**Colombia**
Siemens S.A.
**Bogotá**
**Baranquilla**
**Cali**
**Medellín**

**Costa Rica**
Siemens S.A.
**San José**

**Ecuador**
Siemens S.A.
**Quito**
OTESA
**Guayaquil**
**Quito**

**El Salvador**
Siemens S.A.
**San Salvador**

**Guatemala**
Siemens S.A.
**Ciudad de Guatemala**

**Honduras**
Representaciones Electro-industriales S. de R.L.
**Tegucigalpa**

**Mexico**
Siemens S.A.
**México, D.F.**
**Culiacán**
**Gómez Palacio**
**Guadalajara**
**León**
**Monterrey**
**Puebla**

**Nicaragua**
Siemens S.A.
**Managua**

**Paraguay**
Rieder & Cia., S.A.C.I.
**Asunción**

**Peru**
Siemsa
**Lima**

**Uruguay**
Conatel S.A.
**Montevideo**

**Venezuela**
Siemens S.A.
**Caracas**
**Valencia**

**United States of America**
Siemens Industrial Automation Inc.
**Alpharetta,** Georgia

## Asia

**Bahrain**
Transitec Gulf
**Manama**
or
Siemens Resident Engineer
**Abu Dhabi**

**Bangladesh**
Siemens Bangladesh Ltd.
**Dhaka**

**Hong Kong**
Jebsen & Co., Ltd.
**Hong Kong**

**India**
Siemens India Ltd.
**Bombay**
**Ahmedabad**
**Bangalore**
**Calcutta**
**Madras**
**New Dehli**
**Secundarabad**

**Indonesia**
P.T.Siemens Indonesia
**Jakarta**
P.T. Dian-Graha Elektrika
**Jakarta**
**Bandung**
**Medan**
**Surabaya**

**Iran**
Siemens Sherkate Sahami Khass
**Teheran**

**Iraq**
Samhiry Bros. Co. (W.L.L.)
**Baghdad**
or
Siemens AG (Iraq Branch)
**Baghdad**

**Japan**
Siemens K.K.
**Tokyo**

**Jordan**
Siemens AG (Jordan Branch)
**Amman**
or
A.R. Kevorkian Co.
**Amman**

**Korea** (Republic)
Siemens Electrical Engineering Co., Ltd.
**Seoul**
**Pusan**

**Kuwait**
National & German Electrical and Electronic Service Co. (INGEECO)
**Kuwait,** Arabia

**Lebanon**
Ets. F.A. Kettaneh S.A.
**Beirut**

**Malaysia**
Siemens AG
Malaysian Branch
**Kuala Lumpur**

**Oman**
Waleed Associates
**Muscat**
or
Siemens Resident Engineers
**Dubai**

**Pakistan**
Siemens Pakistan Engineering Co., Ltd.
**Karachi**
**Islamabad**
**Lahore**
**Peshawer**
**Quetta**
**Rawalpindi**

**People's Republic of China**
Siemens Representative Office
**Beijing**
**Guangzhou**
**Shanghai**

**Philippine Islands**
Maschinen & Technik Inc. (MATEC)
**Manila**

**Qatar**
Trags Electrical Engineering and Air Conditioning Co.
**Doha**
or
Siemens Resident Engineer
**Abu Dhabi**

**Saudi Arabia**
Arabia Electric Ltd. (Equipment)
**Jeddah**
**Damman**
**Riyadh**

**Sri Lanka**
Dimo Limited
**Colombo**

**Syria**
Siemens AG (Damascus Branch)
**Damascus**

**Taiwan**
Siemens Liaison Office
**Taipei**
TAI Engineering Co., Ltd.
**Taipei**

**Thailand**
B. Grimm & Co., R.O.P.
**Bangkok**

**United Arab Emirates**
Electro Mechanical Co.
**Abu Dhabi**
or
Siemens Resident Engineer
**Abu Dhabi**
Scientechnic
**Dubai**
or
Siemens Resident Engineer
**Dubai**

## Asia (continued)

**Yemen** (Arab Republic)
Tihama Tractors &
Engineering Co.o., Ltd.
**Sanaa**
or
Siemens Resident Engineer
**Sanaa**

## Australia

**Australia**
Siemens Ltd.
**Melbourne**
**Brisbane**
**Perth**
**Sydney**

**New Zealand**
Siemens Liaison Office
**Auckland**

**Index**

# Index


**A**

Accumulator 8-10, 8-12
Actual operand 7-14
Addition 8-31
Address
- absolute 5-9
- relative 5-10
Address assignment 6-7
- in RAM 6-15
- in the system data area 6-16
AM flag 12-10
Analog input module 11-1, 11-11
Analog modules
- addressing 6-5
Analog output module 11-20
Analog value
- conversion 11-22
- output of (FB251) 9-14, 11-25,
- read in (FB250) 9-14, 11-17, 11-22
- scaling (FB250) 9-14, 11-22
Argument 9-5
Arithmetic operations
- comparison 8-30
- system 8-67
Arithmetic unit 2-5
ASCII mode 15-63
Assigning parameters 9-1
Automation equipment
- fault C-1

**B**

Back-up battery 4-8
BASIC
- creating a program in 15-66
Basic operations 8-1
Battery 4-8
- failure (OB34) 9-14
Binary coded representation (BCD) 7-31, 7-32
Binary divider 8-71
Binary scaler 8-71
Bit pattern 11-11
Bit test operation 8-42
Block
- call operations 8-33
- end symbol 9-4
- header 7-8
- ID 9-1, 9-5, 9-10
- length 7-7
- parameters 7-14
- programming 7-8
- structure 7-6 - 7-8
- type 7-5
Boolean logic operation 8-2
Broken wire 11-7
BSTACK 5-11
Bus cable 13-1
Bus terminal 13-1
Bus unit 2-2
- installing 3-3

**C**

CE marking 14-1
Central Processing Unit (CPU) 2-1, 3-2
Circuit diagram 7-3
Clock data
- 12-hour mode 12-10
- 24-hour mode 12-10
- area 12-8, 12-9, 12-15
- range definition 12-10
Clock pulse generator 8-73
Clock time correction factor 12-7, 12-35
Closed-loop control module 15-41
Code converter
- : 16 9-12
- : B4 9-12
Comment 9-6
- symbol 9-6
Communications module 15-62
- CP 521 Basic 15-65
- CP 521 SI 15-62
Comparator module 15-1
Comparison
- operation 8-30
Complement
- one's 8-50
- two's 8-50
COMPRESS 7-30
Condition code generation 8-69

Control
- deviation 9-21
- system flowchart (CSF) 7-2
- variable 9-12
- word 9-19
Controller
- continuous action 9-15
Controller DB 9-15, 9-19
Conversion operation 8-50
Coordination byte Receive (KBE) 13-2, 13-10
Coordination byte Send (KBS) 13-2, 13-8
Correction rate
- algorithm 9-18
Correction value 12-35
Counter 2-4, 8-26 - 8-29
- loading 8-25
- operation 8-25
- resetting 8-28, 8-29
- scanning 8-26
- setting 8-28, 8-29
Counter module
- 25/500 kHz 15-17
- 2×0 to 500 Hz 15-12
Counting pulse sensor
- connection of 15-21
CPU 2-1
Crimp-snap-in
- connection method 3-10

**D**

Data block 7-5, 7-16
- calling 8-33, 8-35
- deleting 8-33, 8-35
- generating 8-33, 8-35
Data cycle 2-7
- interupt 2-7
Data exchange 13-7
DB1 9-1, 12-2
- function 7-17
DB1 parameter 9-10, 12-2
- transfering 9-9
- setting 9-10
Decimal format 7-32
Default DB1 9-1, 13-5
Derivative action time 9-19, 9-21
Design
- modular 1-2
Diagnostic
- module 15-9
Digital input module 3-13
Digital input/output module 3-18, 6-4
- address assignment 6-7
Digital logic operation 8-44
Display generation operation 8-39
Divider : 16 9-13
DO operation 8-54

**E**

Electromagnetic interference 3-22
Electronic cam controller module 15-49
Enable operation 8-41
Equipotential bonding 3-31
Error
- address 5-9
- analysis 5-1, 5-4, 5-5
- indication 5-1
- remedy 5-4, 5-5
- parameter error 9-4, 9-8
Expansion capability
- maximum 2-8

**F**

Fault
- automation equipment C-1
FB250 11-22, 11-23
FB251 11-25
Field transfer 8-66
Filler 9-1, 9-5
Flags 2-4, 7-3
FORCE VAR 12-19
Formal operand 8-58
Four wire circuit 11-6
Function module
- addressing 6-7
Function block 7-5, 7-11
- calling 7-14
- header 7-12
- integrated 9-11
- setting parameters 7-12, 7-15

**G**

GRAPH 5 7-1
Grounding 3-30

**H**

Hexadecimal representation 7-31

**I**

IM 2-2
IM 315 3-5
IM 316 3-5
Increment operation 8-52
Input 7-3
Input/output module 2-2

Installation of the S5-100U 3-1
- electrical 3-20, 3-21
- horizontally 3-7
- mechanical 3-1
- mechanical, with external I/Os 3-4
- vertical 3-8
Integral action time (TN) 9-19
Interface
- module 2-2, 3-5, 3-6
- serial 2-4
Interrupt
- disable 8-53
- PII 7-29, 10-3
- PIQ 7-29, 10-3
- reaction time 10-7
Interrupt data cycle 2-7
I/O bus 2-5, 2-6, 15-10
I/O modules 3-13, 5-12

**J**
Jump
- processing 8-57
- operation 8-56

**L**
Ladder diagram (LAD) 7-1
Leap year 12-10
Lightning protection 3-30
Linerization 11-8
Load operation 8-10, 8-11, 8-40, 8-64
Loading a time 8-14

**M**
Mode change 7-21
Modular design 1-2
Momentary contact relay/edge evaluation 8-71
Monitoring 5-10
Mounting rail
- standard 2-2
Multiplier: 16 9-16

**N**
Nesting depth 7-6
"NO" operation 8-38
Normal mode 7-19
Number
- format 7-31

**O**
OB2 10-1, 10-4
OB13 7-28
OB21 7-24
OB22 7-24
On-delay 8-22, 8-23
- stored 8-23
- timer 15-6
Operand 7-1
- areas 7-3
- ID 7-1
Operating hours counter 12-7, 12-30
Operating mode
- changing 4-2
- display 4-1
- panel 4-2
- start-up 4-2
- switch 4-1
Operating system 2-7
Operation 7-1, 8-1
- basic 8-1
- Boolean logic 8-2 - 8-7
- set/reset 8-7 - 8-9
- supplementary 8-1
- system 8-1
Organization block 7-5, 7-9, 7-18
- integrated 9-14
Output 7-3
Overall reset 4-2

**P**
Parameter
- function block 7-12
Parameter block 9-1, 9-5
Parameter error 9-4
- correction 9-6
- locating 9-8
- recognizing 9-8
Parameter error code 9-2, 9-7
- scanning 9-6
Parameter name 9-5
PID control algorithm (OB251) 9-15
PM flag 12-10
Position
- decoding 15-29
- resolution 15-19, 15-26, 15-30

Position sensor
- connecting 15-20, 15-2
Positioning 15-57
- algorithm 9-18
- closed-loop controlled 15-60
- open-loop controlled 15-56
Positioning module
- IP 263 15-45
- IP 266 15-55
Power supply
- frequency 11-7
- module 2-1, 3-2, 3-12
3-20
Printer communications module CP521 15-62
Printer mode 15-63
Process image (PII, PIQ) 2-4, 7-29
Process image I/O tables 10-4
- interrupt 6-12
- PII 6-8, 6-10, 10-3
- PIQ 6-8, 6-11, 10-3
Processor 2-5
Program
- block 7-5, 7-11
- memory 2-4, 7-30
- structured 8-33
Program check 4-11
Program processing 7-18
- cyclical 7-26
- interrupt-driven 6-12, 7-29, 10-1
- time-controlled 6-12, 7-28
Programmable controller
- design 2-1
Programming
- linear 7-4
- structured 7-5
Prompt time 12-6, 12-25
Proportional gain 9-19
PT 100 11-6
Pulse generator
- connection of 15-21
Pulse timer 8-20
- extended 8-21

**R**

Reaction time
- interrupt 10-5
Real-time clock
- integral 12-1
- reading 12-21
- setting 12-5, 12-21
Receive Mailbox (EF) 13-2
Reference
- point approach 15-31
- pulse 15-32
- signal 15-32
- variable 9-21
Reference potential 11-1
Register contents
- loading and transfering 8-65
Removing the S5-100U 3-1, 3-2
Response time 7-27
Retentive characteristics 2-5
Retriggering
- OB31 8-33
RLO 8-33

**S**

Sampling interval 9-18, 9-21
Scan cycle time trigger 7-26
Scan monitoring time 7-26
Screw-type 3-18
- connection method 3-9
SEARCH 5-11
- function 4-11
Send Mailbox (SF) 13-2
Sensor lines 11-19
Serial interface 2-4
Sequence block 7-5, 7-11
Set operation 8-64
Set/reset operation 8-7 - 8-9
Setpoint 9-19, 9-21
Set time 15-5
Setting parameters
- for function blocks 7-15
Shielding 3-29
Shift operation 8-48
Shift register 2-6
- length 2-8
Shunt resistor 11-5
Simulator
- module 15-7
SINEC L1
- local area network 13-1

Slave 13-3, 13-5
Slot addressing 6-1
SONAR BERO 11-23
Start ID 9-4, 9-5
START-UP 4-1, 7-24
Starting up 4-4
Statement list (STL) 7-1
STATUS 4-8
STATUS VAR 4-9
Status word 12-12, 12-15
Stepper motor control 15-59
STOP operation 8-39
Substitution operation 8-58
Subtraction operation 8-31
Supplimentary operation 8-1, 8-31
System
- data 6-16
- operations 8-1, 8-64 - 8-67
- parameters 5-14
System data area 12-15
System data word 13-2
System characteristics
- defining in DB1 9-11

**T**

Terminal block 3-10
Temperature 11-3
- compensation 11-8
Test function
- STATUS 4-8
Test mode 7-19
Thermocouples 11-2
Time 8-17
- base 8-16, 8-17
- loading 8-14, 8-17
Time constant
- dominant 9-21
Timer 8-15 - 8-24
- module 15-4
- operation 8-15
- reset 8-15
- starting 8-15, 8-19
Transfer 8-12
- operation 8-10, 8-11, 8-64
Transfering a time 8-14
Two's complement 11-11

**U**

USTACK 5-1

**W**

Wiring
- arrangement 3-29
Wiring method
- crimp-snap-in terminals 3-10
- screw-type terminals 3-9

Siemens AG
A&D AS E 148
Postfach 1963

D-92209 Amberg
Federal Republic of Germany

From:

Your Name: _____

Your Title: _____

Company Name: _____

Street: _____

City, Zip Code: _____

Country: _____

Phone: _____

Please check any industry that applies to you:

- ☐ Automotive
- ☐ Chemical
- ☐ Electrical Machinery
- ☐ Food
- ☐ Instrument and Control
- ☐ Nonelectrical Machinery
- ☐ Petrochemical
- ☐ Pharmaceutical
- ☐ Plastic
- ☐ Pulp and Paper
- ☐ Textiles
- ☐ Transportation
- ☐ Other _____

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Title of Your Manual: _____

Order No. of Your Manual: _____

Edition: _____

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1. Do the contents meet your requirements? ☐
2. Is the information you need easy to find? ☐
3. Is the text easy to understand? ☐
4. Does the level of technical detail meet your requirements? ☐
5. Please rate the quality of the graphics/tables: ☐

Additional comments:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____