

## SIMATIC

### System Software for S7-300/400 System and Standard Functions

#### Reference Manual

C79000-G7076-C503-02

Preface, Contents	
Organization Blocks	1
General Parameters for SFCs	2
Copying and Block Functions	3
SFCs for Program Control	4
SFCs for Handling the Clock	5
SFCs for Handling Run-Time Meters	6
SFCs for Transferring Data Records	7
SFCs for Handling Time-of-Day Interrupts	8
SFCs for Handling Time-Delay Interrupts	9
SFCs for Handling Synchronous Errors	10
SFCs for Handling Interrupts and Asynchronous Errors	11
SFCs for Diagnostics	12
SFCs for Updating the Process Image and Processing Bit Fields	13
SFCs for Addressing Modules	14
SFCs for Distributed I/Os	15
SFCs for Global Data Communication	16
Communication SFBs for Configured Connections	17
Communication SFCs for Non-Configured Connections	18
Generating Block-Related Messages	19
IEC Timers and IEC Counters	20
IEC Functions	21
SFBs for Integrated Control	22
Appendix, Glossary, Index	

## Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:



---

### 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

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

---

## Qualified Personnel

The device/system may only be set up and operated in conjunction with this manual.

Only **qualified personnel** should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

## Correct Usage

Note the following:



---

### Warning

This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.

This product can only function correctly and safely if it is transported, stored, set up, and installed correctly, and operated and maintained as recommended.

---

## Trademarks

SIMATIC® and SINEC® are registered trademarks of SIEMENS AG.

Third parties using for their own purposes any other names in this document which refer to trademarks might infringe upon the rights of the trademark owners.

### Copyright © Siemens AG 1996 All rights reserved

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.

### Disclaimer of Liability

We have checked the contents of this manual for agreement with the hardware and software described. Since deviations cannot be precluded entirely, we cannot guarantee full agreement. However, the data in this manual are reviewed regularly and any necessary corrections included in subsequent editions. Suggestions for improvement are welcomed.

Siemens AG  
Automation Group  
Industrial Automation Systems  
Postfach 4848, D-90327 Nürnberg

Siemens Aktiengesellschaft

Technical data subject to change.

© Siemens AG 1996

C79000-G7076-C503

# Preface

## Purpose

This manual provides you with a comprehensive overview of the organization blocks (OB), system functions (SFC), communications function blocks (CFB), system and standard function blocks (SFC), and IEC functions contained in the operating systems of the CPUs of the S7-300 and S7-400. The appendix describes the diagnostic data, system status lists (SZL), and events.

---

### Note

Refer to the reference section of the “*S7-300 Programmable Controller, Hardware and Installation*” manual /70/ or the “*S7-400/M7-400 Programmable Controllers Module Specifications*” reference manual /101/ or the Instruction List: *S7-400 Programmable Controller* /102/ for details of which of these functions and blocks are available on which CPU. The properties of the CFBs and the S7 signaling functions for specific CPUs are described in /70/ and /101/.

---

For information about the CPU operating systems, program design, and the communications and diagnostic capabilities of the CPUs, refer to the “*System Software for S7-300 and S7-400, Program Design*” programming manual /234/. How to call functions and function blocks in your program is explained in the language descriptions. You program and assign parameters for all these functions using the STEP 7 standard software. How to use this software is described in the “*Standard Software for S7 and M7, STEP 7*” user manual /231/ and in the STEP 7 online help.

## Audience

This manual is intended for programmers and engineers who are familiar with controlling processes and are responsible for writing programs for programmable logic controllers.

## Scope of the Manual

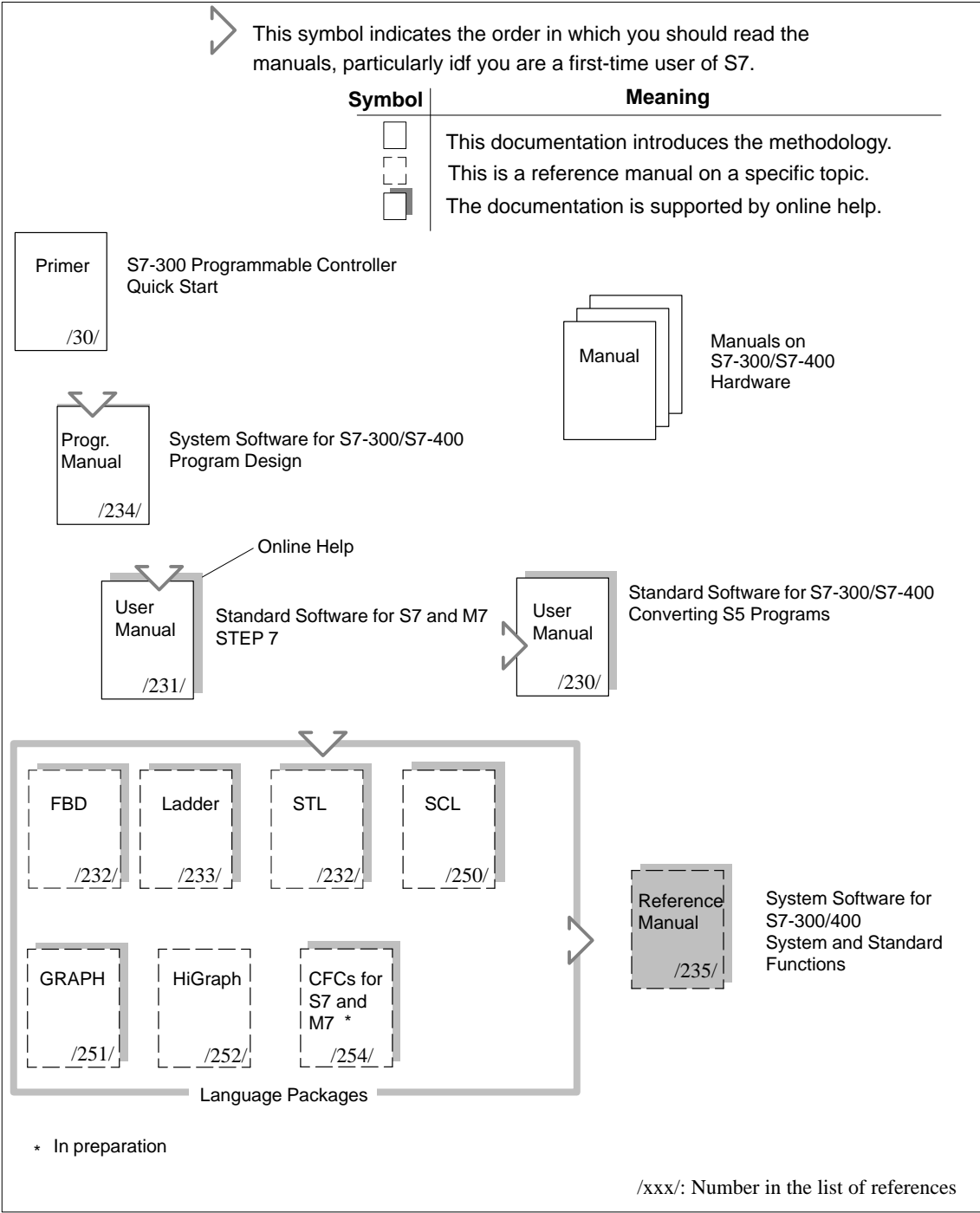
This manual applies to the following CPUs of the S7-300 and S7-400:

CPU	Order Number	Version (or higher)
CPU 312 IFM	6ES7312-5AC01-0AB0	01
CPU 313	6ES7313-1AD01-0AB0	01
CPU 314	6ES7314-1AE02-0AB0	01
CPU 314 IFM	6ES7314-5AE01-0AB0	01
CPU 315	6ES7315-1AF01-0AB0	01
CPU 315-2DP	6ES7315-2AF01-0AB0	01
CPU 412-1	6ES7412-1XF01-0AB0	03
CPU 413-1	6ES7413-1XG01-0AB0	03
CPU 413-2DP	6ES7413-2XG01-0AB0	03
CPU 414-1	6ES7414-1XG01-0AB0	03
CPU 414-2DP	6ES7414-2XG01-0AB0	03
CPU 414-2DP	6ES7414-2XJ00-0AB0	03
CPU 416-1	6ES7416-1XJ01-0AB0	03
CPU 416-2DP	6ES7416-2XK00-0AB0	03
CPU 416-2DP	6ES7416-2XL00-0AB0	03
CPU 614	6ES7614-1AH01-0AB3	01

The CPU functions described in this manual can be used with release 3.1 or higher of the STEP 7 standard software.

Overview of the  
STEP 7  
Documentation

There is a wide range of both general and task-oriented user documentation available to support you when configuring and programming an S7 programmable logic controller. The following descriptions and the figure below will help you to find the user documentation you require.



Title	Subject
<b>S7-300 Programmable Logic Controller Quick Start, Primer</b>	The primer provides you with a very simple introduction to the methods of configuring and programming an S7-300/400. It is particularly suitable for first-time users of an S7 programmable logic controller.
<b>S7-300/400 Program Design Programming Manual</b>	The “ <i>S7-300/400 Program Design</i> ” programming manual provides you with the basic information you require about the structure of the operating system and a user program for an S7 CPU. First-time users of an S7-300/400 should read this manual to get a basic overview of programming methods on which to base the design of a user program.
<b>S7-300/400 System and Standard Functions Reference Manual</b>	The S7 CPUs have system functions and organization blocks integrated in the operating system that can be used when programming in the available languages FBD, LAD, STL and SCL. The manual provides you with an overview of the system functions, organization blocks and loadable standard functions available with an S7 programmable controller and contains detailed interface descriptions explaining how to use the functions and blocks in your user program.
<b>STEP 7 User Manual</b>	The “ <i>STEP 7 User Manual</i> ” explains the basic use and functions of the STEP 7 automation software. Whether you are a first-time user of STEP 7 or an experienced STEP 5 user, the manual will provide you with an overview of the procedures for configuring, programming, and getting started with an S7-300/400 programmable controller. When working with the software, you can call up the online help which supports you with information about specific details of the program.
<b>Converting STEP 5 Programs User Manual</b>	You require the “ <i>Converting STEP 5 Programs User Manual</i> ” if you want to convert existing STEP 5 programs and to run them on S7 CPUs. The manual explains how to use the converter. The online help system provides more detailed information about using the specific converter functions. The online help system also includes an interface description of the available converted S7 functions.
<b>STL, LAD, FBD SCL<sup>1</sup> Manuals</b>	The manuals for the language packages STL, LAD, FBD, and SCL contain both instructions for the user and a description of the language. To program an S7-300/400, you only require one of the languages, but you can, if required, mix the languages within a project. When using one of the languages for the first time, it is advisable to familiarize yourself with the methods of creating a program as explained in the manual. When working with the software, you can use the online help system which provides you with detailed information about using the editors and compilers.
<b>GRAPH<sup>1</sup>, HiGraph<sup>1</sup>, CFC<sup>1</sup> Manuals</b>	The GRAPH, HiGraph, and CFC languages provide you with optional methods for implementing sequential control systems, status control systems, or graphical interconnection of blocks. The manuals contain both the user instructions and the description of the language. When using one of these languages for the first time, it is advisable to familiarize yourself with the methods of creating a program based on the “ <i>S7-300 and S7-400 Program Design</i> ” manual. When working with the software, you can also use the online help system (with the exception of HiGraph) which provides you with detailed information about using the editors and compilers.

<sup>1</sup> Optional packages for the S7-300/400 system software

**Other Manuals**

The various S7-300 and S7-400 CPUs and the S7-300 and S7-400 modules are described in the following manuals:

- For the S7-300 programmable logic controller, refer to the manuals: “*S7-300 Programmable Controller, Hardware and Installation*” /70/, “*S7-300, M7-300 Programmable Controllers Module Specifications*” /71/ and in the Instruction List /72/.
- For the S7-400 programmable logic controller, refer to the manual: “*S7-400/M7-400 Programmable Controllers Module Specifications*” /101/ and in the Instruction List /102/.

**How to Use this Manual**

This manual covers the following topics:

- Chapter 1 explains the functions of all the organization blocks.
- Chapter 2 describes the common parameters RET\_VAL, REQ and BUSY.
- Chapters 3 to 22 describe the SFCs, SFBs and IEC-FCs.
- The Appendix sections A to F contain a description of the structure of the diagnostic data, an overview of the SZL-IDs, the possible events, lists of the SFCs, SFBs and FCs described in this manual, an overview of the SDBs, and a list of references used in the manual.
- The Glossary explains important terminology.
- The Index helps you to locate sections of text and topics quickly.

**Conventions**

References to other manuals and documentation are indicated by numbers in slashes /.../. These numbers refer to the titles of manuals listed in Appendix F.

**Additional Assistance**

If you have any questions regarding the software described in this manual and cannot find an answer here or in the online help, please contact the Siemens representative in your area. You will find a list of addresses in the Appendix of /70/ or /100/, or in catalogs, and in CompuServe (go autforum). You can also speak to our Hotline under the following phone or fax number:

Tel. (+49) (911) 895–7000 (Fax 7001)

If you have any questions or comments on this manual, please fill out the remarks form at the end of the manual and return it to the address shown on the form. We would be grateful if you could also take the time to answer the questions giving your personal opinion of the manual.

Siemens also offers a number of training courses to introduce you to the SIMATIC S7 automation system. Please contact your regional training center or the central training center in Nuremberg, Germany for details:

D–90327 Nuremberg, Tel. (+49) (911) 895–3154.

**Special Note**

The system functions can be interrupted. If there are any restrictions that apply to certain SFCs or situations, these are explained in the description of the particular SFC.





# Contents

	<b>Preface</b> .....	<b>iii</b>
<b>1</b>	<b>Organization Blocks</b> .....	<b>1-1</b>
1.1	Overview of the Organization Blocks (OBs) .....	1-2
	What Are Organization Blocks? .....	1-2
	Which OBs Are Available? .....	1-2
	Where to Find More Information? .....	1-2
1.2	Program Cycle Organization Block (OB1) .....	1-5
	Description .....	1-5
	Understanding the Operation of OB1 .....	1-5
1.3	Time-of-Day Interrupt Organization Blocks (OB10 to OB17) .....	1-7
	Description .....	1-7
	Conditions That Affect Time-of-Day Interrupt OBs .....	1-9
1.4	Time-Delay Interrupt Organization Blocks (OB20 to OB23) .....	1-11
	Description .....	1-11
1.5	Cyclic Interrupt Organization Blocks (OB30 to OB38) .....	1-13
	Description .....	1-13
1.6	Hardware Interrupt Organization Blocks (OB40 to OB47) .....	1-15
1.7	Multicomputing Interrupt Organization Block (OB60) .....	1-17
1.8	Time Error Organization Block (OB80) .....	1-19
	Description .....	1-19
1.9	Power Supply Error Organization Block (OB81) .....	1-21
1.10	Diagnostic Interrupt Organization Block (OB82) .....	1-23
1.11	Insert / Remove Module Interrupt Organization Block (OB83) .....	1-25
1.12	CPU Hardware Fault Organization Block (OB84) .....	1-27
1.13	Priority Class Error Organization Block (OB85) .....	1-28
1.14	Rack Failure Organization Block (OB86) .....	1-31
1.15	Communication Error Organization Block (OB87) .....	1-35
1.16	Background Organization Block (OB90) .....	1-37
1.17	Start-Up Organization Blocks (OB100 and OB101) .....	1-39
1.18	Programming Error Organization Block (OB121) .....	1-42
	Description .....	1-42
	Understanding the Operation of the Programming Error OB .....	1-42
1.19	I/O Access Error Organization Block (OB122) .....	1-45

	Description .....	1-45
	Understanding the Operation of the I/O Access Error OB .....	1-45
<b>2</b>	<b>Common Parameters for SFCs .....</b>	<b>2-1</b>
2.1	Evaluating Errors with the Output Parameter RET_VAL .....	2-2
	Types of Error Information .....	2-2
	Error Information in the Return Value .....	2-2
	Reactions to Error Information .....	2-2
	General and Specific Error Information .....	2-3
	General Error Information .....	2-4
	Specific Error Information .....	2-4
2.2	Meaning of the Parameters REQ, RET_VAL and BUSY with Asynchronous SFCs .....	2-6
	Asynchronous SFCs .....	2-6
<b>3</b>	<b>Copy and Block Functions .....</b>	<b>3-1</b>
3.1	Copying Variables with SFC20 "BLKMOV" .....	3-2
	Description .....	3-2
	Error Information .....	3-3
3.2	Initializing a Memory Area with SFC21 "FILL" .....	3-4
	Description .....	3-4
	Exceptions .....	3-4
	Parameters .....	3-5
	The Input Parameter is a Structure .....	3-5
	Error Information .....	3-5
3.3	Creating a Data Block with SFC22 "CREAT_DB" .....	3-6
	Description .....	3-6
	Interruptability .....	3-6
	Parameters .....	3-6
	Error Information .....	3-7
3.4	Deleting a Data Block with SFC23 "DEL_DB" .....	3-8
	Error Information .....	3-9
3.5	Testing a Data Block with SFC24 "TEST_DB" .....	3-10
3.6	Compressing the User Memory with SFC25 "COMPRESS" .....	3-11
3.7	Transferring a Substitute Value to Accumulator 1 with SFC44 "REPL_VAL" .....	3-13
<b>4</b>	<b>SFCs for Controlling Program Execution .....</b>	<b>4-1</b>
4.1	Retriggering Cycle Time Monitoring with SFC43 "RE_TRIGR" .....	4-2
	Description .....	4-2
	Parameters .....	4-2
	Error Information .....	4-2
4.2	Changing the CPU to STOP with SFC46 "STP" .....	4-3
	Description .....	4-3
	Parameters .....	4-3
	Error Information .....	4-3
4.3	Delaying Execution of the User Program with SFC47 "WAIT" .....	4-4
	Description .....	4-4

	Interruptability .....	4-4
	Parameters .....	4-4
4.4	Triggering a Multicomputing Interrupt With SFC 35 "MP_ALM" .....	4-5
<b>5</b>	<b>SFCs for Handling the System Clock .....</b>	<b>5-1</b>
5.1	Setting the Time with SFC0 "SET_CLK" .....	5-2
	Description .....	5-2
	Date and Time .....	5-2
5.2	Reading the Time with SFC1 "READ_CLK" .....	5-3
	Description .....	5-3
	Parameters .....	5-3
	Error Information .....	5-3
5.3	Synchronizing Slave Clocks with SFC48 "SNC_RTCB" .....	5-4
<b>6</b>	<b>SFCs for Handling Run-Time Meters .....</b>	<b>6-1</b>
6.1	Run-Time Meters .....	6-2
	Introduction .....	6-2
	Application .....	6-2
	Characteristics of the Run-Time Meter .....	6-2
	Range of Values .....	6-2
6.2	Setting the Run-Time Meter with SFC2 "SET_RTM" .....	6-3
	Description .....	6-3
6.3	Starting and Stopping a Run-Time Meter with SFC3 "CTRL_RTM" .....	6-4
	Description .....	6-4
6.4	Reading a Run-Time Meter with SFC4 "READ_RTM" .....	6-5
	Description .....	6-5
	Parameters .....	6-5
6.5	Reading the System Time with SFC64 "TIME_TCK" .....	6-6
	Description .....	6-6
	Application .....	6-6
	System Time and Modes .....	6-6
	Parameters .....	6-6
	Error Information .....	6-6
<b>7</b>	<b>SFCs for Transferring Data Records .....</b>	<b>7-1</b>
7.1	Writing and Reading Data Records .....	7-2
7.2	Writing Dynamic Parameters with SFC55 "WR_PARM" .....	7-4
	Description .....	7-4
	Prior Conditions .....	7-4
	Input Parameter RECORD .....	7-4
7.3	Writing Default Parameters with SFC56 "WR_DPARM" .....	7-5
	Description .....	7-5
7.4	Assigning Parameters to a Module with SFC57 "PARM_MOD" .....	7-6
	Parameters .....	7-6
7.5	Writing a Data Record with SFC58 "WR_REC" .....	7-8
	Description .....	7-8
	Parameters .....	7-8

7.6	Reading a Data Record with SFC59 "RD_REC" .....	7-9
7.7	Reading a Data Record with SFC59 "RD_REC" on S7-300 CPUs .....	7-13
7.8	Further Error Information from SFCs 55 to 59 .....	7-16
<b>8</b>	<b>SFCs for Handling Time-of-Day Interrupts .....</b>	<b>8-1</b>
8.1	Handling Time-of-Day Interrupts .....	8-2
	Definition .....	8-2
	Conditions for the Call .....	8-2
	Tip .....	8-2
	Purpose of SFC28 to SFC31 .....	8-2
8.2	Characteristics of SFCs 28 to 31 .....	8-3
	What Happens If... ..	8-3
	Executing the Time-of-Day Interrupt OBs .....	8-4
8.3	Setting a Time-of-Day Interrupt with SFC28 "SET_TINT" .....	8-5
	Description .....	8-5
8.4	Canceling a Time-of-Day Interrupt with SFC29 "CAN_TINT" .....	8-6
	Description .....	8-6
	Parameters .....	8-6
8.5	Activating a Time-of-Day Interrupt with SFC30 "ACT_TINT" .....	8-7
	Description .....	8-7
8.6	Querying a Time-of-Day Interrupt with SFC31 "QRY_TINT" .....	8-8
	Description .....	8-8
<b>9</b>	<b>SFCs for Handling Time-Delay Interrupts .....</b>	<b>9-1</b>
9.1	Handling Time-Delay Interrupts .....	9-2
	Conditions for the Call .....	9-2
	Purpose of SFC32 to SFC34 .....	9-2
	What Happens if... ..	9-2
	Complete Restart .....	9-3
	Starting in a Start-Up OB .....	9-3
9.2	Starting a Time-Delay Interrupt with SFC32 "SRT_DINT" .....	9-4
	Description .....	9-4
9.3	Querying a Time-Delay Interrupt with SFC34 "QRY_DINT" .....	9-5
	Description .....	9-5
	Parameters .....	9-5
	Output Parameter STATUS .....	9-5
	Error Information .....	9-5
9.4	Canceling a Time-Delay Interrupt with SFC33 "CAN_DINT" .....	9-6
	Description .....	9-6
<b>10</b>	<b>SFCs for Handling Synchronous Errors .....</b>	<b>10-1</b>
10.1	Masking Synchronous Errors .....	10-2
	Example .....	10-6
10.2	Masking Synchronous Errors with SFC36 "MSK_FLT" .....	10-10
	Description .....	10-10
10.3	Unmasking Synchronous Errors with SFC37 "DMSK_FLT" .....	10-11

	Description .....	10-11
	Error Information .....	10-11
10.4	Reading the Error Register with SFC38 "READ_ERR" .....	10-12
	Description .....	10-12
	Parameters .....	10-12
	Error Information .....	10-12
<b>11</b>	<b>SFCs for Handling Interrupts and Asynchronous Errors .....</b>	<b>11-1</b>
11.1	Delaying and Disabling Interrupts and Asynchronous Errors .....	11-2
	Purpose of SFC39 to SFC42 .....	11-2
	Advantage of SFC41 and SFC42 .....	11-2
	Interrupt Classes .....	11-2
	Asynchronous Errors .....	11-3
11.2	Disabling the Processing of New Interrupts and Asynchronous Errors with SFC39 "DIS_IRT" .....	11-4
	Description .....	11-4
	Note .....	11-4
	Parameters .....	11-4
	MODE .....	11-4
	Error Information .....	11-5
11.3	Enabling the Processing of New Interrupts and Asynchronous Errors with SFC40 "EN_IRT" .....	11-6
	Description .....	11-6
	Parameters .....	11-6
	MODE .....	11-6
11.4	Delaying the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC41 "DIS_AIRT" .....	11-8
	Description .....	11-8
	Parameters .....	11-8
	Return Value .....	11-8
11.5	Enabling the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC42 "EN_AIRT" .....	11-9
	Description .....	11-9
	Example .....	11-9
	Parameters .....	11-9
	Return Value and Error Information .....	11-9
<b>12</b>	<b>SFCs for Diagnostics .....</b>	<b>12-1</b>
12.1	Reading OB Start Information with SFC6 "RD_SINFO" .....	12-2
12.2	Reading a System Status List or Partial List with SFC51 "RDSYSST" ...	12-4
	Description .....	12-4
	SZL_HEADER .....	12-5
12.3	Writing a User-Defined Diagnostic Event to the Diagnostic Buffer with SFC52 "WR_USMSG" .....	12-11
	Description .....	12-11
	Sending a User-Defined Diagnostic Message .....	12-11
	Send Buffer Full .....	12-12
	Station Not Logged On .....	12-12
	General Structure .....	12-12

	Event ID .....	12-12
	Additional Information .....	12-12
	Time Stamp .....	12-12
	Parameters .....	12-13
	SEND .....	12-13
	EVENTN .....	12-13
	INFO1 .....	12-13
	INFO2 .....	12-13
<b>13</b>	<b>SFCs for Updating the Process Image and Processing Bit Fields .....</b>	<b>13-1</b>
13.1	Updating the Process Image Input Table with SFC26 "UPDAT_PI" .....	13-2
13.2	Updating the Process Image Output Table with SFC27 "UPDAT_PO" ...	13-3
13.3	Setting a Range of Outputs with SFC79 "SET" .....	13-4
	Error Information .....	13-4
13.4	Resetting a Range of Outputs with SFC80 "RSET" .....	13-5
	Error Information .....	13-5
13.5	Implementing a Sequencer With SFB32 "DRUM" .....	13-6
<b>14</b>	<b>System Functions for Addressing Modules .....</b>	<b>14-1</b>
14.1	Querying the Logical Address of a Channel with SFC5 "GADR_LGC" ...	14-2
14.2	Querying the Module Slot belonging to a Logical Address with SFC49 "LGC_GADR" .....	14-4
14.3	Querying all Logical Addresses of a Module with SFC50 "RD_LGADR" .	14-6
<b>15</b>	<b>SFCs for Distributed I/Os .....</b>	<b>15-1</b>
15.1	Triggering a Hardware Interrupt on the DP Master with SFC7 "DP_PRAL" .....	15-2
15.2	Reading Diagnostic Data of a DP Slave with SFC13 "DPNRM_DG" (Slave Diagnostics) .....	15-4
15.3	Reading Consistent Data of a DP Standard Slave with SFC14 "DPRD_DAT" .....	15-7
15.4	Writing Consistent Data to a DP Standard Slave with SFC15 "DPWR_DAT" .....	15-9
<b>16</b>	<b>SFCs for Global Data Communication .....</b>	<b>16-1</b>
16.1	Sending a GD Packet with SFC60 "GD_SND" .....	16-2
16.2	Fetching a Received GD Packet with SFC61 "GD_RCV" .....	16-4
<b>17</b>	<b>Communication SFBs for Configured Connections .....</b>	<b>17-1</b>
17.1	Classification of the Communication SFBs for Configured Connections ..	17-2
17.2	Classification of the Parameters of Communication SFBs for Configured Connections .....	17-3
17.3	Uncoordinated Sending of Data with SFB8 "USEND" .....	17-7
17.4	Uncoordinated Receiving of Data with SFB9 "URCV" .....	17-8
17.5	Sending Segmented Data with SFB12 "BSEND" .....	17-10

17.6	Receiving Segmented Data with SFB13 "BRCV" .....	17-12
17.7	Read Data from a Remote CPU with SFB14 "GET" .....	17-14
17.8	Writing Data to a Remote CPU with SFB15 "PUT" .....	17-16
17.9	Sending Data to a Printer With SFB 16 "PRINT" .....	17-18
17.10	Initiating a Complete Restart on a Remote Device with SFB 19 "START" .....	17-25
17.11	Changing a Remote Device to the STOP State with SFB20 "STOP" ....	17-27
17.12	Initiating a Restart on a Remote Device with SFB21 "RESUME" .....	17-29
17.13	Querying the Status of a Remote Partner with SFB22 "STATUS" .....	17-31
17.14	Receiving the Status of a Remote Device with SFB23 "USTATUS" .....	17-33
17.15	Querying the Status of the Connection Belonging to a Communication SFB Instance with SFC62 "CONTROL" .....	17-35
17.16	Startup Routine of Communication SFBs for Configured Connections ...	17-37
17.17	How CFBs React to Problems .....	17-39
<b>18</b>	<b>Communication SFCs for Non-Configured Connections .....</b>	<b>18-1</b>
18.1	Differences Compared with Communication SFBs for Configured Connections .....	18-2
18.2	Overview .....	18-4
18.3	Common Parameters of the Communication SFCs .....	18-7
18.4	Data Consistency with GET and PUT SFCs .....	18-8
18.5	Sending Data to a Communication Partner outside the Local S7 Station with SFC65 "X_SEND" .....	18-11
18.6	Receiving Data from a Communication Partner outside the Local S7 Station with SFC66 "X_RCV" .....	18-13
18.7	Reading Data from a Communication Partner outside the Local S7 Station with SFC67 "X_GET" .....	18-17
18.8	Writing Data to a Communication Partner outside the Local S7 Station with SFC68 "X_PUT" .....	18-19
18.9	Aborting an Existing Connection to a Communication Partner outside the Local S7 Station with SFC 69 "X_ABORT" .....	18-21
18.10	Reading Data from a Communication Partner within the Local S7 Station with SFC72 "I_GET" .....	18-22
18.11	Writing Data to a Communication Partner within the Local S7 Station with SFC73 "I_PUT" .....	18-24
18.12	Aborting an Existing Connection to a Communication Partner within the Local S7 Station with SFC 74 "I_ABORT" .....	18-26
18.13	Error Information of the Communication SFCs for Non-Configured Connections .....	18-27

<b>19</b>	<b>Generating Block-Related Messages</b>	<b>19-1</b>
19.1	Introduction to Generating Block-Related Messages With SFBs	19-2
19.2	Generating Block-Related Messages Without Acknowledgment with SFB36 "NOTIFY"	19-4
19.3	Generating Block-Related Messages With Acknowledgment with SFB33 "ALARM"	19-6
19.4	Generating Block-Related Messages with Accompanying Values for Eight Signals with SFB35 "ALARM_8P"	19-9
19.5	Generating Block-Related Messages without Accompanying Values for Eight Signals with SFB34 "ALARM_8"	19-11
19.6	Sending Archive Data with SFB37 "AR_SEND"	19-13
19.7	Disabling Block-Related, Symbol-Related and Group Status Messages with SFC10 "DIS_MSG"	19-15
19.8	Enabling Block-Related, Symbol-Related and Group Status Messages with SFC9 "EN_MSG"	19-17
19.9	Startup Behavior of the SFBs for Generating Block-Related Messages	19-19
19.10	How the SFBs for Generating Block-Related Messages React to Problems	19-20
19.11	Introduction to Generating Block-Related Messages with SFCs	19-21
19.12	Generating Acknowledgable Block-Related Messages with SFC 17 "ALARM_SQ" and Permanently Acknowledged Block-Related Messages with SFC18 "ALARM_S"	19-23
19.13	Querying the Acknowledgment Status of the Last ALARM_SQ Entering Event Message with SFC19 "ALARM_SC"	19-26
<b>20</b>	<b>IEC Timers and IEC Counters</b>	<b>20-1</b>
20.1	Generating a Pulse with SFB3 "TP"	20-2
20.2	Generating an On Delay with SFB4 "TON"	20-3
20.3	Generating an Off Delay with SFB5 "TOF"	20-4
20.4	Counting Up with SFB0 "CTU"	20-5
20.5	Counting Down with SFB1 "CTD"	20-6
20.6	Counting Up and Counting Down with SFB2 "CTUD"	20-7
<b>21</b>	<b>IEC Functions</b>	<b>21-1</b>
21.1	Overview	21-2
21.2	Technical Specifications for the IEC Functions	21-3
21.3	Date and Time as Complex Data Types Actual Parameters for DATE_AND_TIME	21-4
21.4	Date and Time-of-Day Functions: FC3, FC6, FC7, FC8, FC33, FC40, FC1, FC35, FC34	21-5



21.5	Comparing DATE_AND_TIME Variables: FC9, FC12, FC14, FC18, FC23, FC28	21-10
21.6	Comparing STRING Variables: FC10, FC13, FC15, FC19, FC24, FC29	21-13
21.7	Editing STRING Variables: FC21, FC20, FC32, FC26, FC2, FC17, FC4, FC31, FC11	21-16
21.8	Converting Data Type Formats: FC16, FC5, FC30, FC38, FC37, FC39	21-21
21.9	Editing Number Values: FC22, FC25, FC27	21-24
21.10	Binary Selection: FC36	21-26
<b>22</b>	<b>SFBs for Integrated Control</b>	<b>22-1</b>
22.1	Continuous Control with SFB41 "CONT_C"	22-4
22.2	Step Control with SFB42 "CONT_S"	22-11
22.3	Pulse Generation with SFB43 "PULSEGEN"	22-17
22.4	Example of the PULSEGEN Block	22-27
<b>A</b>	<b>Diagnostic Data</b>	<b>A-1</b>
A.1	Overview of the Structure of Diagnostic Data	A-2
	Data Record 0 and 1 of the System Data	A-2
	Structure and Content of the Diagnostic Data	A-2
A.2	Diagnostic Data	A-3
A.3	Structure of Channel-Specific Diagnostic Data	A-5
<b>B</b>	<b>System Status Lists (SZL)</b>	<b>B-1</b>
B.1	Overview of the System Status Lists (SZL)	B-2
	Content	B-2
	System Data	B-2
	Diagnostic Buffer	B-2
B.2	Structure of a Partial SZL List	B-3
	Structure	B-3
	Header	B-3
	Data Records	B-3
B.3	SZL-ID	B-4
	SZL-ID	B-4
	Structure	B-4
	Module Class	B-4
	Number of the Partial List Extract	B-4
	Number of the Partial List	B-4
B.4	Possible Partial System Status Lists	B-5
B.5	SZL-ID W#16#xy00 - List of Available SZL-IDs of a Module	B-6
	Purpose	B-6
	Header	B-6
	Index	B-6
	Data Record	B-6
B.6	SZL-ID W#16#xy11 – Module Identification	B-7
	Purpose	B-7

	Header .....	B-7
	Data Record .....	B-7
B.7	SZL-ID W#16#xy12 – CPU Characteristics .....	B-8
	Purpose .....	B-8
	Header .....	B-8
	Data Record .....	B-8
	Characteristics Identifier .....	B-8
B.8	SZL-ID W#16#xy13 – Memory Areas .....	B-10
	Purpose .....	B-10
	Header .....	B-10
	Data Record .....	B-10
B.9	SZL-ID W#16#xy14 – System Areas .....	B-12
B.10	SZL-ID W#16#xy15 – Block Types .....	B-13
B.11	SZL-ID W#16#xy16 – Existing Priority Classes .....	B-14
B.12	SZL-ID W#16#xy17 – List of Permitted SDBs .....	B-15
	SDB Number .....	B-15
B.13	SZL-ID W#16#xy18 – Maximum S7-300 I/O Configuration .....	B-16
	Purpose .....	B-16
B.14	SZL-ID W#16#xy19 – Status of the Module LEDs .....	B-17
B.15	SZL-ID W#16#xy21 – Assignment of Interrupts and Errors .....	B-18
B.16	SZL-ID W#16#xy22 – Interrupt Status .....	B-20
B.17	SZL-ID W#16#xy23 – Status of the Priority Classes .....	B-22
B.18	SZL-ID W#16#xy24 – Operating Mode and Mode Transition .....	B-24
	Purpose .....	B-24
	Header .....	B-24
	Mode Transition Information .....	B-25
	Mode Transition Information .....	B-27
B.19	SZL-ID W#16#xy31 – Capability Parameters for Communication .....	B-28
B.20	Data Record of the Partial List Extract with SZL-ID W#16#0131 In- dex W#16#0001 .....	B-29
	Content .....	B-29
	Data Record .....	B-29
B.21	Data Record of the Partial List Extract with SZL-ID W#16#0131 In- dex W#16#0002 .....	B-30
	Content .....	B-30
	Data Record .....	B-30
B.22	Data Record of the Partial List Extract with SZL-ID W#16#0131 In- dex W#16#0003 .....	B-32
	Content .....	B-32
	Data Record .....	B-32
B.23	Data Record of the Partial List Extract with SZL-ID W#16#0131 In- dex W#16#0004 .....	B-33
	Content .....	B-33
	Data Record .....	B-33

B.24	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0005	B-34
	Content	B-34
	Data Record	B-34
B.25	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0006	B-35
	Content	B-35
	Data Record	B-35
B.26	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0007	B-37
B.27	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0008	B-38
	Content	B-38
	Data Record	B-38
B.28	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0009	B-39
B.29	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0010	B-40
B.30	Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0011	B-41
B.31	SZL-ID W#16#xy32 – Communication Status Data	B-42
	Data Record	B-42
B.32	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0001	B-43
	Data Record	B-43
B.33	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0002	B-44
	Content	B-44
	Data Record	B-44
B.34	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0003	B-45
	Content	B-45
B.35	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0004	B-46
	Content	B-46
	Data Record	B-46
B.36	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0005	B-47
B.37	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0006	B-48
B.38	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0007	B-49
	Content	B-49
	Data Record	B-49

B.39	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0008 .....	B-50
	Content .....	B-50
	Data Record .....	B-50
B.40	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0009 .....	B-51
	Content .....	B-51
	Data Record .....	B-51
B.41	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#000A .....	B-52
	Content .....	B-52
	Data Record .....	B-52
B.42	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0010 .....	B-53
	Data Record .....	B-53
B.43	Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0011 .....	B-54
	Data Record .....	B-54
B.44	SZL-ID W#16#xy33 – Stations for S7 Messages and Diagnostic Events .	B-55
B.45	SZL-ID W#16#xy81 – Local Data of the OBs .....	B-56
B.46	SZL-ID W#16#xy82 – Startup Events .....	B-57
B.47	SZL-ID W#16#xy91 – Module Status Information .....	B-58
B.48	SZL-ID W#16#xy92 – Rack / Station Status Information .....	B-61
B.49	SZL-ID W#16#xyA0 – Diagnostic Buffer .....	B-64
B.50	SZL-ID W#16#00B1 – Module Diagnostic Information .....	B-65
B.51	SZL-ID W#16#00B2 – Diagnostic Data Record 1 with Geographical Address .....	B-66
B.52	SZL-ID W#16#00B3 – Module Diagnostic Data with Logical Base Address .....	B-67
B.53	SZL-ID W#16#00B4 – Diagnostic Data of a DP Slave .....	B-68
<b>C</b>	<b>Events .....</b>	<b>C-1</b>
C.1	Events and Event ID .....	C-2
	Event .....	C-2
	Event ID .....	C-2
	Event Class .....	C-2
	Identifier .....	C-2
C.2	Event Class 1 – Standard OB Events .....	C-3
C.3	Event Class 2 – Synchronous Errors .....	C-5
C.4	Event Class 3 – Asynchronous Errors .....	C-6
C.5	Event Class 4 – Stop and Abort Events .....	C-8
C.6	Event Class 5 – Mode Run-Time Events .....	C-11

C.7	Event Class 6 – Communication Events .....	C-13
C.8	Event Class 8 – Diagnostic Events for Modules .....	C-15
C.9	Event Class 9 – Standard User Events .....	C-17
C.10	Event Classes A and B – Free User Events .....	C-19
C.11	Reserved Event Classes .....	C-20
<b>D</b>	<b>List of SFCs, SFBs and FCs .....</b>	<b>D-1</b>
D.1	List of SFCs, Sorted Numerically .....	D-2
D.2	List of SFCs, Sorted Alphabetically .....	D-4
D.3	List of SFBs, Sorted Numerically .....	D-6
D.4	List of SFBs, Sorted Alphabetically .....	D-7
D.5	List of FCs .....	D-8
<b>E</b>	<b>System Data Blocks (SDBs) .....</b>	<b>E-1</b>
E.1	System Data Blocks (SDBs) .....	E-2
<b>F</b>	<b>References .....</b>	<b>F-1</b>
	<b>Glossary .....</b>	<b>Glossary-1</b>
	<b>Alphabetical Index .....</b>	<b>Index-1</b>



# Organization Blocks

## Chapter Overview

Section	Description	Page
1.1	Overview of the Organization Blocks (OBs)	1-2
1.2	Program Cycle Organization Block (OB1)	1-5
1.3	Time-of-Day Interrupt Organization Blocks (OB10 to OB17)	1-7
1.4	Time-Delay Interrupt Organization Blocks (OB20 to OB23)	1-11
1.5	Cyclic Interrupt Organization Blocks (OB30 to OB38)	1-13
1.6	Hardware Interrupt Organization Blocks (OB40 to OB47)	1-15
1.7	Multicomputing Interrupt Organization Block (OB60)	1-17
1.8	Time Error Organization Block (OB80)	1-19
1.9	Power Supply Error Organization Block (OB81)	1-21
1.10	Diagnostic Interrupt Organization Block (OB82)	1-23
1.11	Insert/Remove Module Interrupt Organization Block (OB83)	1-25
1.12	CPU Hardware Fault Organization Block (OB84)	1-27
1.13	Priority Class Error Organization Block (OB85)	1-28
1.14	Rack Failure Organization Block (OB86)	1-31
1.15	Communication Error Organization Block (OB87)	1-35
1.16	Background Organization Block (OB90)	1-37
1.17	Start-Up Organization Blocks (OB100 and OB101)	1-39
1.18	Programming Error Organization Block (OB121)	1-42
1.19	I/O Access Error Organization Block (OB122)	1-45

## 1.1 Overview of the Organization Blocks (OBs)

### What Are Organization Blocks?

Organization Blocks (OBs) are the interface between the operating system of the CPU and the user program. OBs are used to execute specific program sections:

- At the startup of the CPU
- In a cyclic or clocked execution
- At specific times of day or on specific days
- After a specified time has elapsed
- Whenever errors occur
- Whenever hardware interrupts occur.

Organization blocks are executed according to the priority they are allocated.

### Which OBs Are Available?

Not all CPUs can process all of the OBs available in STEP 7. Consult the data sheets for your CPU to determine which OBs are included with your CPU.

### Where to Find More Information?

Refer to the online help and the following manuals for more information:

- **/70/**: this manual contains the data sheets that describe the capabilities of the different S7-300 CPUs. This also includes the possible start events for each OB.
- **/101/**: this manual contains the data sheets that describe the capabilities of the different S7-400 CPUs. This also includes the possible start events for each OB.

Table 1-1 contains the start event belonging to each OB as well as the default priority class.



Table 1-1 Overview of the Organization Blocks

OB	Start Event	Default Priority Class	Explanation
OB1	End of startup or end of OB1	1	Free cycle
OB10	Time-of-day interrupt 0	2	No default time specified
OB11	Time-of-day interrupt 1	2	
OB12	Time-of-day interrupt 2	2	
OB13	Time-of-day interrupt 3	2	
OB14	Time-of-day interrupt 4	2	
OB15	Time-of-day interrupt 5	2	
OB16	Time-of-day interrupt 6	2	
OB17	Time-of-day interrupt 7	2	
OB20	Time-delay interrupt 0	3	No default time specified
OB21	Time-delay interrupt 1	4	
OB22	Time-delay interrupt 2	5	
OB23	Time-delay interrupt 3	6	
OB30	Cyclic interrupt 0 (default interval: 5 s)	7	Cyclic interrupts
OB31	Cyclic interrupt 1 (default interval: 2 s)	8	
OB32	Cyclic interrupt 2 (default interval: 1 s)	9	
OB33	Cyclic interrupt 3 (default interval: 500 ms)	10	
OB34	Cyclic interrupt 4 (default interval: 200 ms)	11	
OB35	Cyclic interrupt 5 (default interval: 100 ms)	12	
OB36	Cyclic interrupt 6 (default interval: 50 ms)	13	
OB37	Cyclic interrupt 7 (default interval: 20 ms)	14	
OB38	Cyclic interrupt 8 (default interval: 10 ms)	15	
OB40	Hardware interrupt 0 (default interrupt)	16	Hardware interrupts
OB41	Hardware interrupt 1	17	
OB42	Hardware interrupt 2	18	
OB43	Hardware interrupt 3	19	
OB44	Hardware interrupt 4	20	
OB45	Hardware interrupt 5	21	
OB46	Hardware interrupt 6	22	
OB47	Hardware interrupt 7	23	
OB60	SFC35 "MP_ALM" call	25	Multicomputing interrupt
OB80	Time error	26, 28 <sup>1)</sup>	Asynchronous error interrupts
OB81	Power supply fault	26, 28 <sup>1)</sup>	
OB82	Diagnostic interrupt	26, 28 <sup>1)</sup>	
OB83	Insert/remove-module interrupt	26, 28 <sup>1)</sup>	
OB84	CPU hardware fault	26, 28 <sup>1)</sup>	
OB85	Program error	26, 28 <sup>1)</sup>	
OB86	Failure of an expansion rack, DP master system or station for distributed I/Os	26, 28 <sup>1)</sup>	
OB87	Communication error	26, 28 <sup>1)</sup>	
OB90	Complete restart or delete a block being executed in OB90 or load an OB90 on the CPU or terminate OB90	29 <sup>2)</sup>	Background cycle

Table 1-1 Overview of the Organization Blocks

OB	Start Event	Default Priority Class	Explanation
OB100 OB101	Complete restart Restart	27 <sup>1)</sup> 27 <sup>1)</sup>	} Startup
OB121 OB122	Programming error I/O access error	Priority of the OB causing the error Priority of the OB causing the error	} Synchronous error interrupts

<sup>1)</sup> Priority classes 27 and 28 are valid in the priority class model of the startup.

<sup>2)</sup> Priority class 29 corresponds to priority 0.29. This means that the background cycle has lower priority than the free cycle.

## 1.2 Program Cycle Organization Block (OB1)

**Description** The operating system of the S7 CPU executes OB1 continuously. When OB1 has been executed, the operating system starts it again. Cyclic execution of OB1 is started after the startup has been completed. You can call other function blocks (FBs, SFBs) or functions (FCs, SFCs) in OB1.

**Understanding the Operation of OB1** OB1 has the lowest priority of all of the OBs whose run times are monitored, in other words, all of the other OBs except OB90 can interrupt the execution of OB1. The following events cause the operating system to call OB1:

- The startup is completed.
- The execution of OB1 (the previous cycle) has finished.

When OB1 has been executed, the operating system writes the process-image output table to the output modules and sends any global data. Before restarting OB1, the operating system updates the process-image input table and receives any global data for the CPU.

S7 monitors the maximum scan time, ensuring a maximum response time. The value for the maximum scan time is preset to 150 ms. You can set a new value or you can restart the time monitoring anywhere within your program with SFC43 "RE\_TRIGR". If your program exceeds the maximum cycle time for OB1, the operating system calls OB80 (time error OB); if OB80 is not programmed, the CPU changes to the STOP mode.

Apart from monitoring of the maximum scan time, it is also possible to guarantee a minimum scan time. The operating system will delay the next start of OB1 until the minimum scan time has been reached.

Refer to the manuals **/70/** and **/101/** for the ranges of the parameters "maximum" and "minimum" scan time. You change parameter settings using STEP 7.

**Local Data for OB1** Table 1-2 describes the temporary (TEMP) variables for OB1. The variable names are the default names of OB1.

Table 1-2 Local Variables (TEMP) for OB1

Variable	Type	Description
OB1_EV_CLASS	BYTE	Event class and identifiers: b#16#11 = active
OB1_SCAN_1	BYTE	B#16#01: completion of a complete restart B#16#02: completion of a restart B#16#03: completion of the main cycle
OB1_PRIORITY	BYTE	Priority class 1
OB1_OB_NUMBR	BYTE	OB number (01)
OB1_RESERVED_1	BYTE	Reserved
OB1_RESERVED_2	BYTE	Reserved
OB1_PREV_CYCLE	INT	Run time of previous scan (ms)
OB1_MIN_CYCLE	INT	Minimum cycle time (ms) since the last startup
OB1_MAX_CYCLE	INT	Maximum cycle time (ms) since the last startup
OB1_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

### 1.3 Time-of-Day Interrupt Organization Blocks (OB10 to OB17)

#### Description

STEP 7 provides up to eight OBs (OB10 to OB17) which can be run once or periodically. You can assign parameters for CPU using SFCs or STEP 7 so that these OBs are processed at the following intervals:

- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Annually

#### Understanding the Operation of Time-of-Day Interrupt OBs

To start a time-of-day interrupt, you must first set and then activate the interrupt. The three following start possibilities exist:

- Automatic start of the time-of-day interrupt. This occurs once you have set and then activated the time-of-day interrupt with STEP 7. Table 1-3 shows the basic possibilities for activating a time-of-day interrupt with STEP 7.
- You set the time-of-day interrupt with STEP 7 and then activate it by calling SFC30 “ACT-TINT” in your program.
- You set the time-of-day interrupt by calling SFC28 “SET\_TINT” and then activate it by calling SFC30 “ACT\_TINT”.

Table 1-3 Basic Possibilities for Activating a Time-of-Day Interrupt with STEP 7

Interval	Description
Not activated	The time-of-day interrupt is not executed, even when loaded in the CPU. It can be activated by calling the SFC30.
Activated once only	The time-of-day OB is canceled automatically after it runs the one time specified. Your program can use SFC28 and SFC30 to reset and reactivate the OB.
Activated periodically	When the time-of-day interrupt occurs, the CPU calculates the next start time for the time-of-day interrupt based on the current time of day and the period.

The behavior of the time-of-day interrupt when you move the clock forwards or backwards is described in [/234/](#).

### Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the date and time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, the start date and time, however, are in the past, then the time-of-day interrupt will be processed the next time it is due. This is illustrated in Figure 1-1.

You can disable or delay and re-enable time-of-day interrupts using SFCs 39 to 42. For further information refer to Chapter 11.

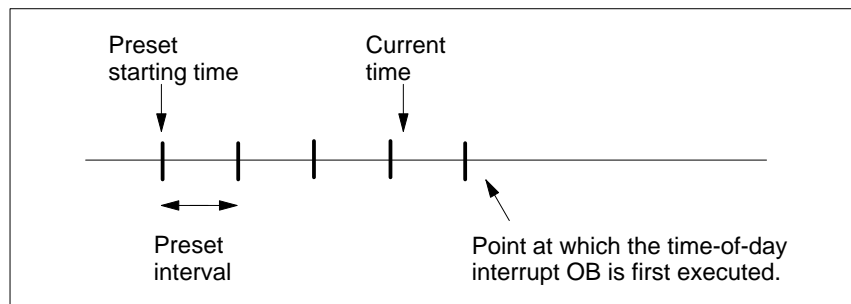


Figure 1-1 First processing of a time-of-day interrupt OB, when the start time is in the past and periodic activation is set.

### Conditions That Affect Time-of-Day Interrupt OBs

Since a time-of-day interrupt occurs only at specified intervals, certain conditions can affect the operation of the OB during the execution of your program. Table 1-4 shows some of these conditions and describes the effect on the execution of the time-of-day interrupt OB.

Table 1-4 Conditions That Affect the Operation of OB10

Condition	Result
Your program calls SFC29 (CAN_TINT) and cancels a time-of-day interrupt.	The operating system clears the start event (date and time) for the considered day-of-time interrupt. You must set the start event again and activate it before the OB can be called again.
Your program attempted to activate a time-of-day interrupt OB, but the OB was not loaded on the CPU.	The operating system calls OB85. If OB85 has not been programmed (loaded on the CPU), the CPU changes to the STOP mode.
When synchronizing or correcting the system clock of the CPU, you set the time ahead and skipped the start event date or time for the time-of-day OB.	The operating system calls OB80 and encodes the number of the time-of-day OB and the start event information in OB80. The operating system then runs the time-of-day OB once, regardless of the number of times that this OB should have been executed. The start event information of OB80 shows the date and time that the time-of-day OB was first skipped.
When synchronizing or correcting the system clock of the CPU, the time was set back so that the start event, date, or time for the OB is repeated.	If the time-of-day OB had already been activated before the clock was set back it is not called again.
The CPU runs through a complete restart.	Any time-of-day OB that was configured by an SFC is changed back to the configuration that was specified in STEP 7.
A time-of-day OB is still being executed when the start event for the next interval occurs.	The operating system calls OB80. If OB80 is not programmed, the CPU changes to the STOP mode. If OB80 is loaded, both OB80 and the time-of-day interrupt OB are first executed and then second the requested interrupt is executed.

**Local Data for  
Time-of-Day  
Interrupt OBs**

Table 1-5 describes the temporary (TEMP) variables for a time-of-day interrupt OB. The variable names are the default names of OB10.

Table 1-5 Local Variables (TEMP) for a Time-of-Day Interrupt OB (here OB10)

Variable	Type	Description
OB10_EV_CLASS	BYTE	Event class and identifiers: b#16#11 = interrupt is active
OB10_STRT_INFO	BYTE	B#16#11: start request for OB10 (B#16#12: start request for OB11) : : (B#16#18: start request for OB17)
OB10_PRIORITY	BYTE	Priority class: 2
OB10_OB_NUMBR	BYTE	OB number (10 to 17)
OB10_RESERVED_1	BYTE	Reserved
OB10_RESERVED_2	BYTE	Reserved
OB10_PERIOD_EXE	WORD	The OB is executed at the specified intervals: W#16#0000: once W#16#0201: once every minute W#16#0401: once hourly W#16#1001: once daily W#16#1201: once weekly W#16#1401: once monthly W#16#1801: once yearly
OB10_RESERVED_3	INT	Reserved
OB10_RESERVED_4	INT	Reserved
OB10_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called



## 1.4 Time-Delay Interrupt Organization Blocks (OB20 to OB23)

### Description

S7 provides up to four OBs (OB20 to OB23) which are executed after a specified delay. Every time-delay OB is started by calling SFC32 (SRT\_DINT). The delay time is an input parameter of the SFC.

When your program calls SFC32 (SRT\_DINT), you provide the OB number, the delay time and a user-specific identifier. After the specified delay, the OB starts. You can also cancel the execution of a time-delay interrupt that has not yet started.

### Understanding the Operation of Time-Delay Interrupt OBs

After the delay time has expired (value in milliseconds transferred to SFC32 together with an OB number), the operating system starts the corresponding OB.

To use the time-delay interrupts, you must perform the following tasks:

- You must call SFC32 (SRT\_DINT).
- You must download the time-delay interrupt OB to the CPU as part of your program.

Time-delay OBs are executed only when the CPU is in the RUN mode. A complete restart clears any start events for the time-delay OBs. If a time-delay interrupt has not started, you can use SFC33 (CAN\_DINT) to cancel its execution.

The delay time has a resolution of 1 ms. A delay time that has expired can be started again immediately. You can query the status of a delay-time interrupt using SFC34 (QRY\_DINT).

The operating system calls an asynchronous error OB if one of the following events occur:

- If the operating system attempts to start an OB that is not loaded and you specified its number when calling SFC32 "SRT\_DINT".
- If the next start event for a time-delay interrupt occurs before the time-delay OB has been completely executed.

You can disable or delay and re-enable delay interrupts using SFCs 39 to 42. For further information refer to Chapter 11.

**Local Data for  
Time-Delay  
Interrupt OBs**

Table 1-6 describes the temporary (TEMP) variables for a time-delay interrupt OB. The variable names are the default names of OB20.

Table 1-6 Local Variables (TEMP) for a Time-Delay Interrupt OB (here OB20)

Variable	Type	Description
OB20_EV_CLASS	BYTE	Event class and identifiers: B#16#11: interrupt is active
OB20_STRT_INF	BYTE	B#16#21: start request for OB20 (B#16#22: start request for OB21) (B#16#23: start request for OB22) (B#16#24: start request for OB23)
OB20_PRIORITY	BYTE	Priority class: 3 (OB20) to 6 (OB23) (default)
OB20_OB_NUMBR	BYTE	OB number (20 to 23)
OB20_RESERVED_1	BYTE	Reserved
OB20_RESERVED_2	BYTE	Reserved
OB20_SIGN	WORD	User ID: input parameter SIGN from the call for SFC32 (SRT_DINT)
OB20_DTIME	TIME	Elapsed delay time in ms
OB20_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

## 1.5 Cyclic Interrupt Organization Blocks (OB30 to OB38)

### Description

S7 provides up to nine cyclic interrupt OBs (OB35 to OB38) which interrupt your program at fixed intervals. Table 1-7 shows the default intervals and priority classes for the cyclic interrupt OBs.

Table 1-7 Default Intervals and Priority Classes for Cyclic Interrupt OBs

OB Number	Default Interval	Default Priority Class
OB30	5 s	7
OB31	2 s	8
OB32	1 s	9
OB33	500 ms	10
OB34	200 ms	11
OB35	100 ms	12
OB36	50 ms	13
OB37	20 ms	14
OB38	10 ms	15

### Understanding the Operation of Cyclic Interrupt OBs

The equidistant start times of the cyclic interrupt OBs are determined by the interval and the phase offset. Refer to [/234/](#) for the relationship between the start time, time cycle, and phase offset of an OB.

#### Note

You must make sure that the run time of each cyclic interrupt OB is significantly shorter than its interval. If a cyclic interrupt OB has not been completely executed before it is due for execution again because the interval has expired, the time error OB (OB80) is started. The cyclic interrupt that caused the error is executed later.

You can disable or delay and re-enable cyclic interrupts using SFCs 39 to 42. For further information refer to Chapter 13.

Refer to the specifications of your specific CPU for the range of the parameters interval, priority class, and phase offset. You can change the parameter settings using STEP 7.

**Local Data for  
Cyclic Interrupt  
OBs**

Table 1-8 describes the temporary (TEMP) variables for a cyclic interrupt OB. The variable names are the default names of OB35.

Table 1-8 Local Variables (TEMP) for a Cyclic Interrupt OB (here OB35)

Variable	Type	Description
OB35_EV_CLASS	BYTE	Event class and identifiers B#16#11: interrupt is active
OB35_STRT_INF	BYTE	(B#16#31: start request for OB30) : B#16#36 : start request for OB35 : (B#16#39: start request for OB38)
OB35_PRIORITY	BYTE	Priority class: 7 (OB30) to 15 (OB38) (default)
OB35_OB_NUMBR	BYTE	OB number (30 to 38)
OB35_RESERVED_1	BYTE	Reserved
OB35_RESERVED_2	BYTE	Reserved
OB35_PHASE_OFFSET	WORD	Phase offset [ms]
OB35_RESERVED_3	INT	Reserved
OB35_EXC_FREQ	INT	Interval in milliseconds
OB35_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

## 1.6 Hardware Interrupt Organization Blocks (OB40 to OB47)

### Description

S7 provides up to eight independent hardware interrupts each with its own OB.

By assigning parameters with STEP 7, you specify the following for each signal module that will trigger hardware interrupts:

- Which channels trigger a hardware interrupt under what conditions.
- Which hardware interrupt OB is assigned to the individual groups of channels (as default, all hardware interrupts are processed by OB40).

With CPs and FMs, you assign these parameters using their own software.

You select the priority classes for the individual hardware interrupt OBs using STEP 7.

### Understanding the Operation of Hardware Interrupt OBs

After a hardware interrupt has been triggered by the module, the operating system identifies the slot and the corresponding hardware interrupt OB. If this OB has a higher priority than the currently active priority class, it will be started. The channel-specific acknowledgement is sent after this hardware interrupt OB has been executed.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then the new interrupt is lost. This is illustrated in Figure 1-2 based on the example of a channel of a digital input module. The triggering event is the rising edge. The hardware interrupt OB is OB40.

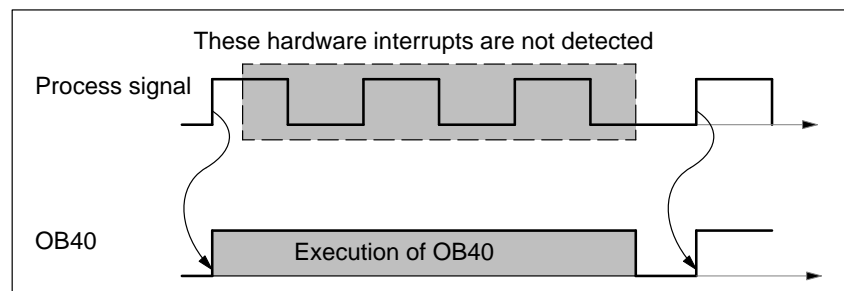


Figure 1-2 Relationship between a Process Signal and the Execution of the Hardware Interrupt OB

- If the event occurs on another channel of the same module, then no hardware interrupt can currently be triggered. This interrupt, however, is not lost, but is triggered after the acknowledgement of the currently active hardware interrupt (only on the S7-400). In the case of the S7-300, the hardware interrupt is lost if the interrupt event is no longer present after the acknowledgement.

If a hardware interrupt is triggered and its OB is currently active due to a hardware interrupt from another module, the new request is recorded and the OB processed when it is free (only on the S7-400). In the case of the S7-300, the hardware interrupt is lost if the interrupt event is no longer present after the acknowledgement.

You can disable or delay and re-enable hardware interrupts using SFCs 39 to 42. For further information refer to Chapter 11.

You can assign parameters for the hardware interrupts of a module not only with STEP 7 but also with SFCs 55 to 57. For further information refer to Chapter 11.

### Local Data for Hardware Interrupt OBs

Table 1-9 describes the temporary (TEMP) variables for a hardware interrupt OB. The variable names are the default names of OB40.

Table 1-9 Local Variables (TEMP) for a Hardware Interrupt OB (here OB40)

Variable	Type	Description
OB40_EV_CLASS	BYTE	Event class and identifiers: B#16#11: interrupt is active
OB40_STRT_INF	BYTE	B#16#41: interrupt via interrupt line 1 B#16#42: interrupt via interrupt line 2 (only with an S7-400) B#16#43: interrupt via interrupt line 3 (only with an S7-400) B#16#44: interrupt via interrupt line 4 (only with an S7-400)
OB40_PRIORITY	BYTE	Priority class: 16 (OB40) to 23 (OB47) (default)
OB40_OB_NUMBR	BYTE	OB number (40 to 47)
OB40_RESERVED_1	BYTE	Reserved
OB40_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB40_MDL_ADDR	WORD	Logical base address of the module that triggers the interrupt
OB40_POINT_ADDR	DWORD	For digital modules: bit field with the statuses of the inputs on the module For analog modules (CP or FM): interrupt status of the module
OB40_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

## 1.7 Multicomputing Interrupt Organization Block (OB60)

### Description

Using the multicomputing interrupt, you can make sure that the reaction of the CPUs is synchronized to an event during multicomputing. In contrast to hardware interrupts triggered by signal modules, the multicomputing interrupt can only be output by CPUs.

### Understanding the Operation of Multicomputing Interrupt OBs

A multicomputing interrupt is triggered by calling SFC35 "MP\_ALM". During multicomputing, this brings about a synchronized OB60 start on all CPUs of the bus segment unless you have disabled OB60 (with SFC39 "DIS\_IRT") or delayed it (with SFC41 "DIS\_AIRT"). If you have not loaded OB60 on a CPU, the CPU returns to the last priority class before the interrupt and continues program execution there. In single processor operation and when using segmented racks, OB60 is only started on the CPU on which you called SFC35 "MP\_ALM".

When your program calls SFC35 "MP\_ALM", you supply a job ID. This ID is transferred to all CPUs. This allows you to react to a specific event. If you program OB60 differently on the various CPUs, this may result in different execution times for the OB. In this case, the CPUs return to the interrupted priority class at different times. If the next multicomputing interrupt is output by a CPU while another CPU is still busy executing the OB60 of the previous multicomputing interrupt, then OB60 is not started either on the requesting or on any other CPU belonging to the bus segment. This is illustrated in Figure 1-3 taking the example of two CPUs. You are informed of the outcome by the function value of the called SFC35.

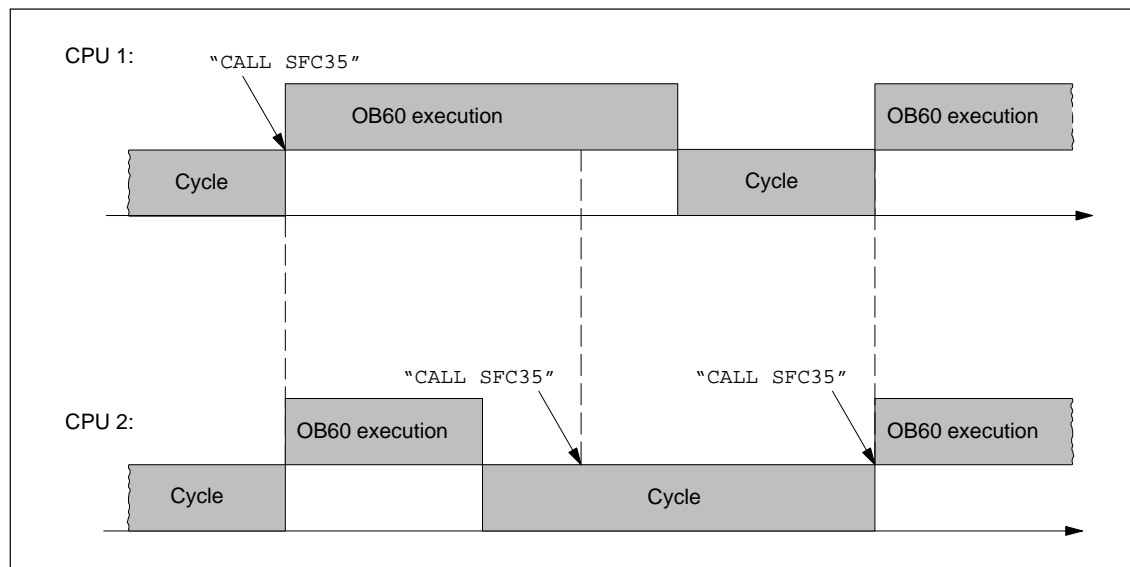


Figure 1-3 Example of Losing a Multicomputing Interrupt

**Local Data for  
Multicomputing  
Interrupt OBs**

Table 1-10 describes the temporary (TEMP) variables of the multicomputing interrupt OB. The variable names are the default names of OB60

Table 1-10 Local Variables (TEMP) of the Multicomputing Interrupt OB

Variable	Data Type	Description
OB60_EV_CLASS	BYTE	Event class and IDs: B#16#11: Interrupt is active
OB60_STRT_INF	BYTE	B#16#61: Multicomputing interrupt triggered by own CPU B#16#62: Multicomputing interrupt triggered by another CPU
OB60_PRIORITY	BYTE	Priority class: 25
OB60_OB_NUMBR	BYTE	OB number: 60
OB60_RESERVED_1	BYTE	Reserved
OB60_RESERVED_2	BYTE	Reserved
OB60_JOB	INT	Job ID: input variable JOB of SFC35 “MP_ALM”
OB60_RESERVED_3	INT	Reserved
OB60_RESERVED_4	INT	Reserved
OB60_DATE_TIME	DATE_AND_TIME	Date and time of day at which the OB was called.



## 1.8 Time Error Organization Block (OB80)

### Description

The operating system of the S7-300 CPU calls OB80 whenever an error occurs while executing an OB. Such errors include: exceeding the cycle time, an acknowledgement error when executing an OB, moving the time forward so that the start time for the OB is skipped. If, for example, a start event for a cyclic OB occurs while the same OB is still being executed following a previous call, the operating system calls OB80.

If OB80 has not been programmed, the CPU changes to the STOP mode.

You can disable or delay and re-enable the time error OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Note

If OB80 is called twice during the same scan cycle due to the scan time being exceeded, the CPU changes to the STOP mode. You can prevent this by calling SFC43 "RE\_TRIGR" at a suitable point in the program.

### Local Data for the Time Error OB

Table 1-11 describes the temporary (TEMP) variables for the time error OB. The variable names are the default names of OB80.

Table 1-11 Local Variables (TEMP) for OB80

Variable	Type	Description
OB80_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB80_FLT_ID	BYTE	Error code: (possible values B#16#01, B#16#02, B#16#05, B#16#06 or B#16#07)
OB80_PRIORITY	BYTE	Priority class: 26
OB80_OB_NUMBR	BYTE	OB number (80)
OB80_RESERVED_1	BYTE	Reserved
OB80_RESERVED_2	BYTE	Reserved
OB80_ERROR_INFO	WORD	Error information: depends on the error code
OB80_ERR_EV_CLASS	BYTE	Event class for the start event that caused the error
OB80_ERR_EV_NUM	BYTE	Event number for the start event that caused the error
OB80_OB_PRIORITY	BYTE	Priority class of the OB active when the error occurred
OB80_OB_NUM	BYTE	Number of the OB that was active when the interrupt occurred
OB80_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

The variables dependent on the error code have the following meaning:

- Error code B#16#01 Cycle time exceeded.  
 OB80\_ERROR\_INFO: Run time of last scan cycle (ms).  
 OB80\_ERR\_EV\_CLASS: Class of the event that triggered the interrupt.  
 OB80\_ERR\_EV\_NUM: Number of the event that triggered the interrupt.  
 OB80\_OB\_PRIORITY: Priority class active at the time of the interrupt.  
 OB80\_OB\_NUM: OB active at the time of the interrupt.
- Error code B#16#02 The OB requested is still being executed.  
 B#16#07 Overflow of OB request buffer for the current priority class (each OB start request for a priority class will be entered in the corresponding OB request buffer; after completion of the OB the entry will be deleted.  
  
 If there are more OB start requests for a priority class than the maximum permitted number of entries in the corresponding OB request buffer, OB80 will be called with error code B#16#07).  
 OB80\_ERROR\_INFO: Like the corresponding value in the interrupted priority class.  
 OB80\_ERR\_EV\_CLASS: Class of the event that triggered the interrupt.  
 OB80\_ERR\_EV\_NUM: Number of the event that triggered the interrupt.  
 OB80\_OB\_PRIORITY: Priority class active at the time of the interrupt.  
 OB80\_OB\_NUM: OB active at the time of the interrupt.
- Error code B#16#05 Elapsed time-of-day interrupt due to moving the clock forward.  
 B#16#06 Elapsed time-of-day interrupt on return to RUN after HOLD.  
 OB80\_ERROR\_INFO: Bit 0 set: The start time for time-of-day interrupt 0 is in the past.  
 :  
 :  
 Bit 7 set: The start time for time-of-day interrupt 7 is in the past.  
 Bit 8 to 15: not used  
 OB80\_ERR\_EV\_CLASS: not used  
 OB80\_ERR\_EV\_NUM: not used  
 OB80\_OB\_PRIORITY: not used  
 OB80\_OB\_NUM: not used

## 1.9 Power Supply Error Organization Block (OB81)

### Description

The operating system of the S7-300 CPU calls OB81 whenever an event occurs that is triggered by an error or fault related to the power supply (only on an S7-400) or the back-up battery (when entering and when leaving the state).

In contrast to the other asynchronous error OBs, the CPU does not change to the STOP mode if OB81 is not programmed.

You can disable or delay and re-enable the power supply error OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Local Data for the Power Supply Error OB

Table 1-12 describes the temporary (TEMP) variables for the power supply error OB. The variable names are the default names of OB81.

Table 1-12 Local Variables (TEMP) for OB81

Variable	Type	Description
OB81_EV_CLASS	BYTE	Event class and identifiers: B#16#38: entering the state B#16#39: leaving the state
OB81_FLT_ID	BYTE	Error code: (possible values B#16#21, B#16#22, B#16#23, B#16#31, B#16#32 or B#16#33)
OB81_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB81_OB_NUMBR	BYTE	OB number (81)
OB81_RESERVED_1	BYTE	Reserved
OB81_RESERVED_2	BYTE	Reserved
OB81_MDL_ADDR	INT	Reserved
OB81_RESERVED_3	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_4	BYTE	
OB81_RESERVED_5	BYTE	
OB81_RESERVED_6	BYTE	
OB81_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

The variables OB81\_RESERVED\_i,  $3 \leq i \leq 6$  indicate the expansion racks on which the battery backup (error code B#16#31), the back-up voltage (error code B#16#32) or the 24-V power supply (error code B#16#33) has failed or returned. The following table shows what bit is assigned to which expansion rack in the variables OB81\_RESERVED\_i,  $3 \leq i \leq 6$ .

Table 1-13

	OB81_RESERVED_6	OB81_RESERVED_5	OB81_RESERVED_4	OB81_RESERVED_3
Bit 0	Reserved	8th expansion rack	16th expansion rack	Reserved
Bit 1	1st expansion rack	9th expansion rack	17th expansion rack	Reserved
Bit 2	2nd expansion rack	10th expansion rack	18th expansion rack	Reserved
Bit 3	3rd expansion rack	11th expansion rack	19th expansion rack	Reserved
Bit 4	4th expansion rack	12th expansion rack	20th expansion rack	Reserved
Bit 5	5th expansion rack	13th expansion rack	21st expansion rack	Reserved
Bit 6	6th expansion rack	14th expansion rack	Reserved	Reserved
Bit 7	7th expansion rack	15th expansion rack	Reserved	Reserved

The bits in the variables OB81\_RESERVED\_i have the following meaning (for the expansion rack concerned):

When the event occurs, the expansion racks are marked (the corresponding bits are set) on which at least one battery or back-up voltage or the 24 V power supply has failed. Expansion racks on which at least one battery or back-up voltage or the 24 V power supply failed earlier are no longer indicated.

When the event is eliminated and the backup is restored on at least one expansion rack, this is signaled (the corresponding bits are set).

The variable OB81\_FLT\_ID has the following meaning:

- B#16#21: At least one back-up battery of the central rack is exhausted/problem eliminated (BATTF)
- B#16#22: Back-up voltage in the central rack failed/problem eliminated (BAF)
- B#16#23: Failure of the 24 V power supply in the central rack/problem eliminated.
- B#16#31: At least one back-up battery of at least one expansion rack is exhausted/problem eliminated (BATTF).
- B#16#32: Back-up voltage in at least one expansion rack not found/problem eliminated (BAF)
- B#16#33: Failure of the 24 V power supply in at least one expansion rack/problem eliminated.

## 1.10 Diagnostic Interrupt Organization Block (OB82)

### Description

If a module with diagnostic capability for which you have enabled the diagnostic interrupt detects an error, it outputs a request for a diagnostic interrupt to the CPU (when entering and leaving the state). The operating system then calls OB82.

The local variables of OB82 contain the logical base address as well as four bytes of diagnostic data of the defective module (see Table 1-14).

If OB82 has not been programmed, the CPU changes to the STOP mode.

You can disable or delay and re-enable the diagnostic interrupt OB using SFCs 39 to 42. For further information refer to Chapter 13.

### Local Data for Diagnostic Interrupt OB

Table 1-14 describes the temporary (TEMP) variables for the diagnostic interrupt OB. The variable names are the default names of OB82.

Table 1-14 Local Variables (TEMP) for OB82

Variable	Type	Description
OB82_EV_CLASS	BYTE	Event class and identifiers: B#16#38: leaving the state B#16#39: entering the state
OB82_FLT_ID	BYTE	Error code (B#16#42)
OB82_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB82_OB_NUMBR	BYTE	OB number (82)
OB82_RESERVED_1	BYTE	Reserved
OB82_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB82_MDL_ADDR	INT	Logical base address of the module where the fault occurred
OB82_MDL_DEFECTION	BOOL	Module is defective
OB82_INT_FAULT	BOOL	Internal fault
OB82_EXT_FAULT	BOOL	External fault
OB82_PNT_INFO	BOOL	Channel fault
OB82_EXT_VOLTAGE	BOOL	External voltage failed
OB82_FLD_CONNCTR	BOOL	Front panel connector not plugged in
OB82_NO_CONFIG	BOOL	Module is not configured
OB82_CONFIG_ERR	BOOL	Incorrect parameters on module
OB82_MDL_TYPE	BYTE	Bit 0 to 3: Module class Bit 4: Channel information exists Bit 5: User information exists Bit 6: Diagnostic interrupt from substitute Bit 7: Reserve

Table 1-14 Local Variables (TEMP) for OB82, continued

Variable	Type	Description
OB82_SUB_MDL_ERR	BOOL	Submodule is missing or has an error
OB82_COMM_FAULT	BOOL	Communication problem
OB82_MDL_STOP	BOOL	Operating mode (0: RUN, 1: STOP)
OB82_WTCH_DOG_FLT	BOOL	Watchdog timer responded
OB82_INT_PS_FLT	BOOL	Internal power supply failed
OB82_PRIM_BATT_FLT	BOOL	Battery exhausted
OB82_BCKUP_BATT_FLT	BOOL	Entire backup failed
OB82_RESERVED_2	BOOL	Reserved
OB82_RACK_FLT	BOOL	Expansion rack failure
OB82_PROC_FLT	BOOL	Processor failure
OB82_EPROM_FLT	BOOL	EPROM fault
OB82_RAM_FLT	BOOL	RAM fault
OB82_ADU_FLT	BOOL	ADC/DAC error
OB82_FUSE_FLT	BOOL	Fuse tripped
OB82_HW_INTR_FLT	BOOL	Hardware interrupt lost
OB82_RESERVED_3	BOOL	Reserved
OB82_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

## 1.11 Insert / Remove Module Interrupt Organization Block (OB83)

### Description

The insertion and removal of modules is monitored within the system at intervals of one second. For the CPU to recognize that a module has been removed and inserted you must wait a minimum of two seconds between removing and inserting.

Each time a configured module is removed or inserted during the RUN, STOP, and start-up modes, an insert/remove interrupt is generated (power supply modules, CPUs, adapter modules and IMs must not be removed in these modes). This interrupt causes an entry in the diagnostic buffer and in the system status list for the CPU involved. The insert/remove OB is also started if the CPU is in the RUN mode. If this OB has not been programmed, the CPU changes to the STOP mode.

You can disable or delay and re-enable the insert/remove OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Understanding the Operation of OB83

If you remove a configured module in the RUN mode, OB83 is started. Since the existence of modules is only monitored at intervals of one second, an access error may be detected first if the module is accessed directly or when the process image is updated.

If you insert a module in a configured slot in the RUN mode, the operating system checks whether the type of the module inserted corresponds to the recorded configuration. OB83 is then started and parameters are assigned if the module types match.

### Local Data for OB83

Table 1-15 describes the temporary (TEMP) variables for the insert/remove module interrupt OB. The variable names are the default names of OB83.

Table 1-15 Local Variables (TEMP) for OB83

Variable	Type	Description
OB83_EV_CLASS	BYTE	Event class and identifiers: B#16#38: module inserted B#16#39: module removed or not responding
OB83_FLT_ID	BYTE	Error code: (possible values B#16#61, B#16#63 or B#16#64)
OB83_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB83_OB_NUMBR	BYTE	OB number (83)
OB83_RESERVED_1	BYTE	Reserved
OB83_MDL_TD	BYTE	Range: B#16#54: Peripheral input (PI) B#16#55: Peripheral output (PQ)
OB83_MDL_ADDR	WORD	Logical base address of the module
OB83_RACK_NUM	WORD	Rack number or number of the DP station (low byte) and DP master system ID (high byte)
OB83_MDL_TYPE	WORD	Module type of the module affected
OB83_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

The variable OB83\_MDL\_TYPE dependent on the error code has the following meaning:

- Error code B#16#61: Module inserted. Module type OK  
(for event class B#16#38)  
  
Module removed or not responding  
(for event class B#16#39)  
  
OB83\_MDL\_TYPE: Actual module type
- Error code B#16#63: Module inserted  
but incorrect module type  
  
OB83\_MDL\_TYPE: Actual module type
- Error code B#16#64: Module inserted  
but problem (type ID cannot be read)  
  
OB83\_MDL\_TYPE: Configured module type
- Error code B#16#65: Module inserted  
but error in module parameter assignment  
  
OB83\_MDL\_TYPE: Actual module type



## 1.12 CPU Hardware Fault Organization Block (OB84)

### Description

The CPU's operating system calls OB84 whenever an error is detected on the interface to the MPI network, to the K bus, or to the interface module for the distributed I/Os.

If OB84 has not been programmed, the CPU changes to the STOP mode when this type of error is detected.

You can disable or delay and re-enable the CPU hardware error OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Local Data for the Hardware Fault OB

Table 1-16 includes the temporary (TEMP) variables of the CPU hardware fault. The variable names are the default names of OB84.

Table 1-16 Local Variables (TEMP) for OB84

Variable	Type	Description
OB84_EV_CLASS	BYTE	Event class and identifiers: B#16#38: leaving the state B#16#39: entering the state
OB84_FLT_ID	BYTE	Error code (B#16#81)
OB84_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB84_OB_NUMBR	BYTE	OB number (84)
OB84_RESERVED_1	BYTE	Reserved
OB84_RESERVED_2	BYTE	Reserved
OB84_RESERVED_3	WORD	Reserved
OB84_RESERVED_4	DWORD	Reserved
OB84_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

## 1.13 Priority Class Error Organization Block (OB85)

### Description

The operating system of the CPU calls OB85 whenever one of the following events occurs:

- Start event for an OB that has not been loaded.
- Error when the operating system accesses a module.
- I/O access error when updating the entire process image.

---

### Note

If OB85 has not been programmed, the CPU changes to the STOP mode when one of these errors is detected.

---

You can disable or delay and re-enable the priority class error OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Local Data for the Priority Class Error OB

Table 1-17 describes the temporary (TEMP) variables for the priority class error OB. The variable names are the default names of OB85.

Table 1-17 Local Variables (TEMP) for OB85

Variable	Type	Description
OB85_EV_CLASS	BYTE	Event class and identifiers: B#16#35, B#16#39 (only with error codes B#16#B1 and B#16#B2)
OB85_FLT_ID	BYTE	Error code: B#16#A1, B#16#A2, B#16#A3, B#16#B1, or B#16#B2
OB85_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB85_OB_NUMBR	BYTE	OB number (85)
OB85_RESERVED_1	BYTE	Reserved
OB85_RESERVED_2	BYTE	Reserved
OB85_RESERVED_3	INT	Reserved
OB85_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB85_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB85_OB_PRIOR	BYTE	Priority class of the OB that was active when the error occurred
OB85_OB_NUM	BYTE	Number of the OB that was active when the error occurred
OB85_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

If you want to program OB85 dependent on the possible error codes, we recommend that you organize the local variables as follows:

Table 1-18 Local Variables (TEMP) for OB85

Variable	Type
OB85_EV_CLASS	BYTE
OB85_FLT_ID	BYTE
OB85_PRIORITY	BYTE
OB85_OB_NUMBR	BYTE
OB85_DKZ23	BYTE
OB85_RESERVED_2	BYTE
OB85_Z1	WORD
OB85_Z23	DWORD
OB85_DATE_TIME	DATE_AND_TIME

The variables modified compared with the default have the following meaning, dependent on the error code:

- Error code B#16#A1 As a result of your configuration created with STEP 7, your program or the operating system creates a start event for an OB that is not loaded on the CPU.
- B#16#A2 As a result of your configuration created with STEP 7, your program or the operating system creates a start event for an OB that is not loaded on the CPU.
- OB85\_Z1: Similar to the corresponding value in the interrupted priority class
- OB85\_Z23: high word: class and number of the event causing the OB call  
low word: program level and OB active at the time of error
- Error code B#16#A3 Error when the operating system accesses a module.
- OB85\_Z1: Error ID of the operating system  
high byte: 1: integrated function  
2: IEC timer  
low byte: 0: no error resolution  
1: block not loaded  
2: area length error  
3: write-protect error

- OB85\_Z23                    high word:   block number  
                                 low word:   relative address  
                                 of the MC7 command causing the error.  
                                 The block type may be taken from OB85\_DKZ23  
                                 (B#16#88: OB, B#16#8C: FC, B#16#8E: FB,  
                                 B#16#8A: DB).
- Error code    B#16#B1: I/O access error during the updating of  
                                 the entire process input image.  
                                 B#16#B2: I/O access error during the updating of  
                                 the entire process output image.
- OB85\_Z1:    Reserved for internal use by the CPU.  
OB85\_Z23:   Number of the I/O byte causing the I/O access error.

## 1.14 Rack Failure Organization Block (OB86)

### Description

The operating system of the CPU calls OB86 whenever the failure of an expansion rack, a DP master system, or a station is detected in the distributed I/Os (both when entering and leaving the state).

If OB86 has not been programmed, the CPU changes to the STOP mode when this type of error is detected.

You can disable or delay and re-enable OB86 using SFCs 39 to 42. For further information refer to Chapter 11.

### Local Data for the Rack Failure OB

Table 1-19 describes the temporary (TEMP) variables for the rack failure OB. The variable names are the default names of OB86.

Table 1-19 Local Variables (TEMP) for OB86

Variable	Type	Description
OB86_EV_CLASS	BYTE	Event class and identifiers: B#16#38: leaving the state B#16#39: entering the state
OB86_FLT_ID	BYTE	Error code: (possible values B#16#C1, B#16#C2, B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7)
OB86_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB86_OB_NUMBR	BYTE	OB number (86)
OB86_RESERVED_1	BYTE	Reserved
OB86_RESERVED_2	BYTE	Reserved
OB86_MDL_ADDR	WORD	Depends on the error code
OB86_RACKS_FLTD	Array [0 ..31] of BOOL	Depends on the error code
OB86_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

If you want to program OB86 dependent on the possible error codes, we recommend that you organize the local variables as follows:

Table 1-20 Local Variables (TEMP) for OB86

Variable	Type
OB86_EV_CLASS	BYTE
OB86_FLT_ID	BYTE
OB86_PRIORITY	BYTE
OB86_OB_NUMBR	BYTE
OB86_RESERVED_1	BYTE
OB86_RESERVED_2	BYTE
OB86_MDL_ADDR	WORD
OB86_Z23	DWORD
OB86_DATE_TIME	DATE_AND_TIME

The variables whose content is dependent on the error code have the following meaning:

- Error code B#16#C1: Expansion rack failure

OB86\_MDL\_ADDR: Logical base address of the IM

OB86\_Z23: Contains one bit for each possible expansion rack:

Bit 0:	always 0
Bit 1:	1st expansion rack
:	
:	
Bit 21:	21st expansion rack
Bits 22 to 29:	Always 0
Bit 30:	Failure of at least one expansion rack in the SIMATIC S5 area
Bit 31:	Always 0

Meaning: when the event occurs, the expansion racks that caused OB86 to be called are reported as having failed (the bits assigned to them are set). Expansion racks that had already failed earlier are no longer indicated. When the failure is eliminated, the expansion racks that are active again are reported in the error code (the bits assigned to them are set).

- Error code B#16#C2: Expansion rack returned with discrepancy between expected and actual configuration)
  - OB86\_MDL\_ADDR: Logical base address of the IM
  - OB86\_Z23: Contains one bit for every possible expansion rack, see error code B#16#C1.  
Meaning of a bit when set (for the expansion rack concerned)
    - modules with an incorrect type ID exist
    - configured modules missing
    - at least one module is defective.
- Error code B#16#C3:
  - Distributed I/Os: Failure of the master system. (Only entering the state causes the start of OB86 with error code B#16#C3. Leaving the state starts OB86 with the error code B#16#C4 and event class B#16#38. The return of every DP slave station starts OB86.)
  - OB86\_MDL\_ADDR: Logical base address of the DP master.
  - OB86\_Z23: DP master system ID
    - Bit 0 to 7: reserved
    - Bit 8 to 15: DP master system ID
    - Bit 16 to 31: reserved
- Error code B#16#C4: Failure of a DP station.  
B#16#C5: Distributed I/Os: station problem.
  - OB86\_MDL\_ADDR: Logical base address of the DP master.
  - OB86\_Z23: Address of DP slave concerned:
    - Bits 0 to 7: no. of DP station
    - Bits 8 to 15: DP master system ID
    - Bits 16 to 30: logical base address of an S7 slave or diagnostic address of a standard DP slave
    - Bit 31: I/O identifier

- Error code B#16#C6: Expansion rack operational again but error in module parameter assignment

OB86\_MDL\_ADDR: Logical base address of IM

OB86\_Z23: Contains a bit for every possible expansion rack:

Bit 0:	always 0
Bit 1:	1st expansion rack
:	
:	
Bit 21:	21st expansion rack
Bit 22 to 30:	reserved
Bit 31:	always 0

Meaning when bit set (in expansion rack concerned):

- modules with incorrect type identifiers exist
- modules with missing or incorrect parameters exist

- Error code B#16#C7: DP: station operational again but error in module parameter assignment

OB86\_MDL\_ADDR: Logical base address of the DP master

OB86\_Z23: Address of the DP slave affected:

Bit 0 to 7:	No. of the DP station
Bit 8 to 15:	DP master system ID
Bit 16 to 30:	Logical base address of the DP slave
Bit 31:	I/O identifier



## 1.15 Communication Error Organization Block (OB87)

### Description

The operating system of the CPU calls OB87 whenever an event occurs that was caused by a communication error.

If OB87 has not been programmed, the CPU changes to the STOP mode when this type of error is detected.

You can disable or delay and re-enable the communication error OB using SFCs 39 to 42. For further information refer to Chapter 11.

### Local Data for OB87

Table 1-21 describes the temporary (TEMP) variables for the communication error OB. The variable names are the default names of OB87.

Table 1-21 Local Variables (TEMP) for OB87

Variable	Type	Description
OB87_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB87_FLT_ID	BYTE	Error code: (possible values B#16#D2, B#16#D3, B#16#D4, B#16#D5, B#16#E1, B#16#E2, B#16#E3, B#16#E4, B#16#E5, or B#16#E6)
OB87_PRIORITY	BYTE	Priority class: 26 (default for the RUN mode) or 28 (STARTUP mode)
OB87_OB_NUMBR	BYTE	OB number (87)
OB87_RESERVED_1	BYTE	Reserved
OB87_RESERVED_2	BYTE	Reserved
OB87_RESERVED_3	WORD	Depends on the error code
OB87_RESERVED_4	DWORD	Depends on the error code
OB87_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

The variables dependent on the error code have the following meaning:

- Error code B#16#D2: Transmission of diagnostic entries currently not possible.  
B#16#D3: Synchronization messages cannot be transmitted (master).  
B#16#D4: Illegal time-of-day jump due to clock synchronization.  
B#16#D5: Error when receiving synchronization time (slave).  
OB87\_RESERVED\_3: Contains no further information.  
OB87\_RESERVED\_4: Contains no further information.



## 1.16 Background Organization Block (OB90)

### Description

With STEP 7, you can monitor a maximum scan cycle time and can guarantee a minimum scan cycle time. If the execution time of OB1 including all the nested interrupts and system activities is less than the minimum scan cycle time that you have specified, the operating system reacts as follows:

- It calls the background OB (providing it exists on the CPU).
- It delays the next OB1 start (if OB90 does not exist on the CPU).

### Understanding the Operation of OB 90

OB90 has the lowest priority of all OBs. It is interrupted by any system activity and any interrupt (even by OB1 after the minimum cycle time has elapsed) and is only resumed if the selected minimum scan cycle time has not yet been reached. The one exception to this is the execution of SFCs that are started in OB90. These are executed with the priority of OB1 and are therefore not interrupted by OB1. There is no time monitoring of OB90.

The user program in OB90 is processed starting with the first instruction in the following situations:

- Following a complete restart or restart
- After deleting a block being executed in OB90 (with STEP 7)
- After loading OB90 on the CPU in the RUN mode
- After terminating the background cycle

## Local Data for OB90

Table 1-22 describes the temporary (TEMP) variables of OB90. The variable names are the default names of OB90.

Table 1-22 Local Variables (TEMP) of OB90

Variable	Data Type	Description
OB90_EV_CLASS	BYTE	Event class and identifiers: B#16#11: active
OB90_STRT_INF	BYTE	B#16#91: complete restart/restart B#16#92: block deleted B#16#93: loading OB90 on the CPU in the RUN mode B#16#95: termination of the background cycle
OB90_PRIORITY	BYTE	Priority class: 29
OB90_OB_NUMBR	BYTE	OB number (90)
OB90_RESERVED_1	BYTE	Reserved
OB90_RESERVED_2	BYTE	Reserved
OB90_RESERVED_3	INT	Reserved
OB90_RESERVED_4	INT	Reserved
OB90_RESERVED_5	INT	Reserved
OB90_DATE_TIME	DATE_AND_TIME	Date and time of day at which the OB was called

## 1.17 Start-Up Organization Blocks (OB100 and OB101)

### Types of Start-Up

- With the S7-300 the only start-up mode available is a complete restart.
- With the S7-400 there are two start-up modes available: complete restart and restart.

During the startup, the operating system calls the start-up OB for the particular start-up mode: for a complete restart this is the complete restart OB (OB100) and for a restart this is the restart OB (OB101). Refer to [/234/](#) for further information about the activities during startup.

### Description

The CPU executes a start-up as follows:

- After POWER ON
- Whenever the mode selector is switched from STOP to RUN
- After a request using a communication function (menu command from the programming device or by calling the communication function blocks “START” or “RESUME” on a different CPU).

A complete restart or a restart OB will be called depending on the start event, the CPU involved, and the parameter settings. You can program specific default settings for your cyclic program.

### Local Data for Start-Up OBs

Table 1-23 describes the temporary (TEMP) variables for a start-up OB. The variable names are the default names of OB100.

Table 1-23 Local Variables (TEMP) for a Start-Up OB (here OB100)

Variable	Type	Description
OB100_EV_CLASS	BYTE	Event class and identifiers: B#16#13: active
OB100_STRTUP	BYTE	Start-up request: B#16#81: manual complete restart B#16#82: automatic complete restart B#16#83: manual startup B#16#84: automatic startup
OB100_PRIORITY	BYTE	Priority class: 27
OB100_OB_NUMBR	BYTE	OB number (100)
OB100_RESERVED_1	BYTE	Reserved
OB100_RESERVED_2	BYTE	Reserved
OB100_STOP	WORD	Number of the event that caused the CPU to stop
OB100_STRT_INFO	DWORD	Supplementary information about the current startup (see Table 1-24)
OB100_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

Table 1-24 Bits in the Variables OB100\_STRT\_INFO and OB101\_STRT\_INFO, continued

Bit No.	Meaning	Possible Binary Values	Explanation
31 - 28	Multiprocessor behavior	0000 0001 0010	Single-processor operation Multicomputing (only S7-400) Operation of several CPUs in a segmented rack (only S7-400)
27 - 24	Additional start-up ID	...0 ...1 ..0. ..1. 0... 1...	No difference in configured/actual configuration (only S7-300) Difference in configured/actual configuration (only S7-300) No difference in configured/actual configuration Difference in configured/actual configuration Clock for time stamp not backed up during last POWER ON Clock for time stamp backed up during last POWER ON
23 - 16	Start-up type just executed	0000 0001 0000 0011 0000 0100 0000 1010 0000 1011 0000 1100 0001 0000 0001 0011 0001 0100 0010 0000 0010 0011 0010 0100 1010 0000	Complete restart in multicomputing without command to CPU according to parameter assignment (only S7-400) Complete restart set at mode selector Complete restart triggered via MPI Restart in multicomputing without command to CPU according to parameter assignment (only S7-400) Restart set at mode selector (only S7-400) Restart triggered via MPI (only S7-400) Automatic complete restart after battery-backed POWER ON Complete restart set at mode selector; last POWER ON battery backed Complete restart triggered via MPI; last POWER ON battery backed Automatic complete restart after POWER ON without battery backup (with system memory reset) Complete restart set at mode selector; last POWER ON not battery backed Complete restart triggered via MPI; last POWER ON not battery backed Automatic restart after battery-backed POWER ON according to the parameter assignment (only S7-400)
15 - 12	Permissibility of automatic startup types	0000 0001 0111 1111	Automatic startup not permitted, memory reset requested Automatic startup not permitted, parameter settings etc. must be changed Complete automatic restart permitted Complete automatic restart / restart permitted (only S7-400)

Table 1-24 Bits in the Variables OB100\_STRT\_INFO and OB101\_STRT\_INFO, continued

Bit No.	Meaning	Possible Binary Values	Explanation
11 - 8	Permissibility of manual start-up types	0000 0001 0111 1111	Startup not permitted, memory reset required Startup not permitted, parameter settings etc. must be changed Complete restart permitted Complete restart and restart permitted (only S7-400)
7 - 0	Last valid operator intervention or setting of the automatic start-up type at POWER ON	0000 0000 0000 0001  0000 0011 0000 0100 0000 1010  0000 1011 0000 1100 0001 0000 0001 0011  0001 0100  0010 0000 0010 0011 0010 0100  1010 0000	No start-up type Complete restart in multicomputing without command to CPU according to parameter assignment (only S7-400) Complete restart triggered by switch setting Complete restart triggered via MPI Restart in multicomputing without command to CPU according to parameter assignment (only S7-400) Restart set at mode selector (only S7-400) Restart triggered via MPI (only S7-400) Automatic complete restart after battery-backed POWER ON Complete restart set at mode selector; last POWER ON with battery backup Complete restart triggered via MPI; last POWER ON with battery backup Automatic complete restart after POWER ON without battery backup (with system memory reset) Complete restart set at mode selector; last POWER ON without battery backup Complete restart triggered via MPI last POWER ON without battery backup Automatic restart after battery-backed POWER ON according to parameter assignment (only S7-400)

## 1.18 Programming Error Organization Block (OB121)

<b>Description</b>	The operating system of the CPU calls OB121 whenever an event occurs that is caused by an error related to the processing of the program. For example, if your program calls a block that has not been loaded on the CPU, OB121 is called.
<b>Understanding the Operation of the Programming Error OB</b>	<p>OB121 is executed in the same priority class as the interrupted block.</p> <p>If OB121 is not programmed, the CPU changes from the RUN mode to the STOP mode.</p> <p>S7 provides the following SFCs for masking and unmasking start events for OB121 during the execution of your program:</p> <ul style="list-style-type: none"><li>• SFC36 (MSK_FLT): masks specific error codes</li><li>• SFC37 (DMSK_FLT): unmask the error codes that were masked by SFC36</li><li>• SFC38 (READ_ERR): reads the error register</li></ul>



### Local Data for the Programming Error OB

Table 1-25 describes the temporary (TEMP) variables for the programming error OB. The variable names are the default names of OB121.

Table 1-25 Local Variables (TEMP) for OB121

Variable	Type	Description
OB121_EV_CLASS	BYTE	Event class and identifiers: B#16#25
OB121_SW_FLT	BYTE	Error code : (possible values B#16#21, B#16#22, B#16#23, B#16#24, B#16#25, B#16#26, B#16#27, B#16#28, B#16#29, B#16#30, B#16#31, B#16#32, B#16#33, B#16#34, B#16#35, B#16#3A, B#16#3C, B#16#3D, B#16#3E, or B#16#3F)
OB121_PRIORITY	BYTE	Priority class = priority class of the OB in which the error occurred
OB121_OB_NUMBR	BYTE	OB number (121)
OB121_BLK_TYPE	BYTE	Type of block where the error occurred (no valid value is entered here in case of S7-300): B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB
OB121_RESERVED_1	BYTE	Reserved
OB121_FLT_REG	WORD	Source of the error (depends on error code). For example: <ul style="list-style-type: none"> <li>Register where the conversion error occurred</li> <li>Incorrect address (for read/write error)</li> <li>Incorrect timer-, counter, or block number</li> <li>Incorrect area identification</li> </ul>
OB121_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error
OB121_PRG_ADDR	WORD	Relative address of the MC7 command that caused the error (no valid value is entered here for an S7-300)
OB121_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called

The variables dependent on the error code have the following meaning:

- Error code B#16#21: BCD conversion error  
OB121\_FLT\_REG: ID for the register concerned (W#16#0000: accumulator 1)
- Error code B#16#22: Area length error when reading  
B#16#23: Area length error when writing  
B#16#28: Read access to a byte, word, or double word with a pointer whose bit address is not 0.  
B#16#29: Write access to a byte, word, or double word with a pointer whose bit address is not 0.

- OB121\_FLT\_REG: Incorrect byte address. The data area and access type can be read from OB121\_RESERVED\_1.
- OB121\_RESERVED\_1: Bits 7 to 4 access type.  
 0: bit access, 1: byte access, 2: word access,  
 3: double word access
- Bits 3 to 0 memory area:  
 0: I/O area,  
 1: process-image input table,  
 2: process-image output table,  
 3: bit memory, 4: global DB, 5: instance DB,  
 6: own local data, 7: local data of caller
- Error code B#16#24: Range error when reading  
 B#16#25: Range error when writing
  - OB121\_FLT\_REG: Contains the ID of the illegal area in the low byte (B#16#86 of own local data area)
  - Error code B#16#26: Error for timer number  
 B#16#27: Error for counter number
  - OB121\_FLT\_REG: Illegal number
  - Error code B#16#30: Write access to a write-protected global DB  
 B#16#31: Write access to a write-protected instance DB  
 B#16#32: DB number error accessing a global DB  
 B#16#33: DB number error accessing an instance DB
  - OB121\_FLT\_REG: Illegal DB number
  - Error code B#16#34: FC number error in FC call  
 B#16#35: FB number error in FB call  
 B#16#3A: Access to a DB that has not been loaded;  
 the DB number is in the permitted range  
 B#16#3C: Access to an FC that has not been loaded;  
 the FC number is in the permitted range  
 B#16#3D: Access to an SFC that has not been loaded;  
 the SFC number is in the permitted range  
 B#16#3E: Access to an FB that has not been loaded;  
 the FB number is in the permitted range  
 B#16#3F: Access to an SFB that has not been loaded;  
 the SFB number is in the permitted range
  - OB121\_FLT\_REG: Illegal number

## 1.19 I/O Access Error Organization Block (OB122)

### Description

The operating system of the CPU calls OB122 whenever an error occurs while accessing data on a module. For example, if the CPU detects a read error when accessing data on an I/O module, the operating system calls OB122.

### Understanding the Operation of the I/O Access Error OB

OB122 is executed in the same priority class as the interrupted OB. If OB122 is not programmed, the CPU changes from the RUN mode to the STOP mode.

S7 provides the following SFCs for masking and unmasking start events for OB122 during the execution of your program:

- SFC36 (MSK\_FLT): masks specific error codes
- SFC37 (DMSK\_FLT): unmasks the error codes that were masked by SFC36
- SFC38 (READ\_ERR): reads the error register

### Local Data for the I/O Access Error OB

Table 1-26 describes the temporary (TEMP) variables for the I/O access error OB. The variable names are the default names of OB122.

Table 1-26 Local Variables (TEMP) for OB122

Variable	Type	Description
OB122_EV_CLASS	BYTE	Event class and identifiers: B#16#29
OB122_SW_FLT	BYTE	Error code B#16#42 For S7-300: I/O access error, reading For S7-400: error during the first read access after an error occurred B#16#43 For S7-300: I/O access error, writing For S7-400: error during the first write access after an error occurred B#16#44 (only for S7-400) error during the n-th ( $n > 1$ ) read access after an error has occurred B#16#45 (only for S7-400) error during the n-th ( $n > 1$ ) write access after an error has occurred
OB122_PRIORITY	BYTE	Priority class = priority of the OB where the error occurred
OB122_OB_NUMBR	BYTE	OB number (122)
OB122_BLK_TYPE	BYTE	Type of block where the error occurred (B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB)
OB122_MEM_AREA	BYTE	Memory area and access type: Bit 7 to 4 access type: 0: bit access, 1: byte access, 2: word access, 3: double word access Bits 3 to 0 memory area: 0: I/O area, 1: process-image input, 2: process-image output
OB122_MEM_ADDR	WORD	Memory address where the error occurred
OB122_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error.
OB122_PRG_ADDR	WORD	Relative address of the MC7 command that caused the error.
OB122_DATE_TIME	DATE_AND_TIME	Date and time of day when the OB was called



# Common Parameters for SFCs

# 2

## Chapter Overview

Section	Description	Page
2.1	Evaluating Errors with the Output Parameter RET_VAL	2-2
2.2	Meaning of the Parameters REQ, RET_VAL and BUSY with Asynchronous SFCs	2-6

## 2.1 Evaluating Errors with the Output Parameter RET\_VAL

### Types of Error Information

A system function (SFC) executed in your user program, indicates whether or not the CPU was able to execute the function of the SFC successfully. You can obtain information about any errors that occurred in two ways:

- In the BR bit of the status word
- In the output parameter RET\_VAL (return value)

---

#### Note

Before evaluating the output parameters specific to an SFC, you should always follow the steps below:

- First, evaluate the BR bit of the status word.
- Then check the output parameter RET\_VAL.

If the BR bit indicates that an error has occurred or if RET\_VAL contains a general error code, you must not evaluate the SFC output parameter!

---

### Error Information in the Return Value

A system function (SFC) indicates that an error occurred during its execution by entering the value “0” in the binary result bit (BR) of the status word. Some system functions provide an additional error code at an output known as the return value (RET\_VAL) output. If a general error is entered in the output parameter RET\_VAL (see below for explanation), this is only indicated by the value “0” in the BR bit of the status word.

The return value is of the data type integer (INT). The relationship of the return value to the value “0” indicates whether or not an error occurred during execution of the function.

Table 2-1 Return Value Indicates Execution of the Function with or without Error

CPU Execution of the SFC	BR	Return Value	Sign of the Integer
with error(s)	0	less than “0”	negative (sign bit is “1”)
without error	1	greater than or equal to “0”	positive (sign bit is “0”)

### Reactions to Error Information

There are two different types of error code in RET\_VAL as follows:

- A general error code, that all system functions can output and
- A specific error code, that the system function can output and which relates to its specific function.

You can write your program so that it reacts to the errors that occur during execution of a system function. This way you prevent further errors occurring as a result of the first error.

### General and Specific Error Information

The return value (RET\_VAL) of a system function provides one of the two following types of error codes:

- A general error code, that relates to errors that can occur in any system function.
- A specific error code, that relates only to the particular system function.

Although the data type of the output parameter RET\_VAL is integer (INT), the error codes for system functions are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this manual, then display the error code in hexadecimal format.

Figure 2-1 shows the structure of a system function error code in hexadecimal format.

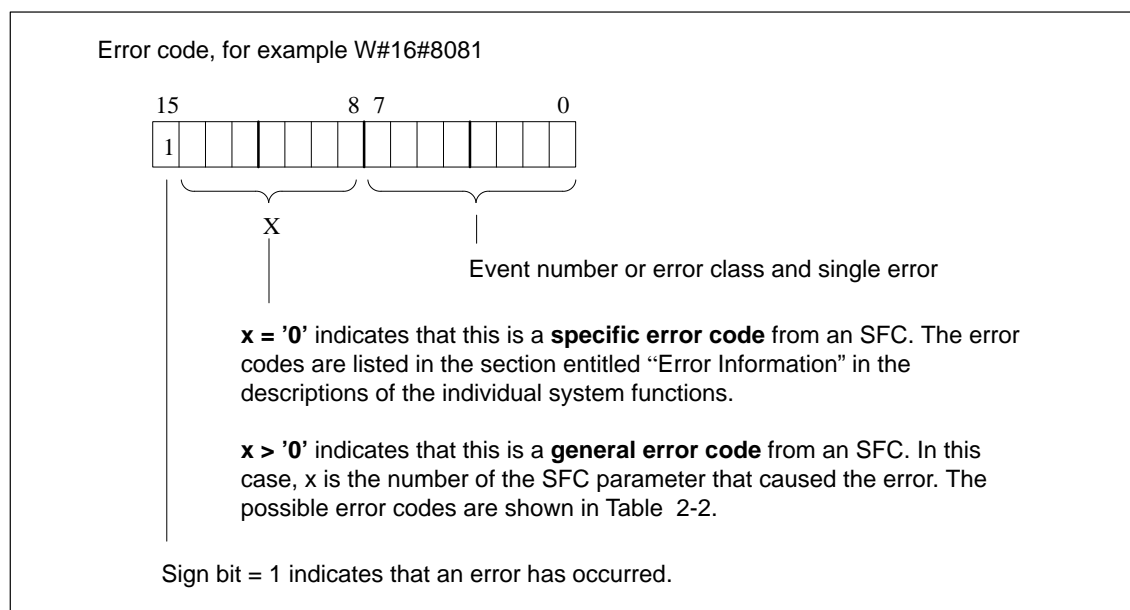


Figure 2-1 Structure of a System Function Error Code in Hexadecimal Format

### General Error Information

The general error code indicates errors that can occur in any system function. A general error code consists of the following two numbers:

- A parameter number from 1 to 127, where 1 indicates the first parameter, 2 indicates the second parameter of the SFC, etc.
- An event number from 0 to 127. The event number indicates that a synchronous error occurred.

Table 2-2 lists the codes for general errors and an explanation of each error.

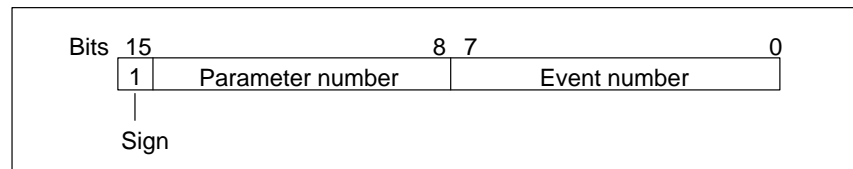


Figure 2-2 Structure of the Return Value Integer in a General Error Code

### Note

If a general error code was entered in RET\_VAL the following situations are possible:

- The action associated with the SFC may have been started or already completed.
- A specific SFC error may have occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

### Specific Error Information

Some system functions (SFCs) have a return value that provides a specific error code. This error code indicates that an error pertaining to a particular system function occurred during the execution of the function (see Figure 2-3). A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.

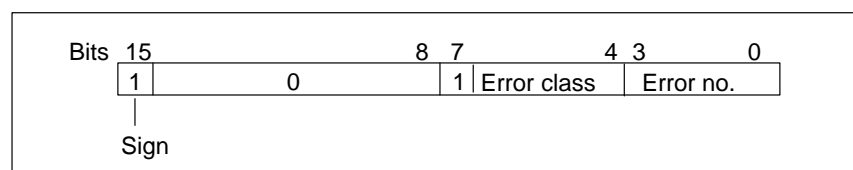


Figure 2-3 Structure of the Integer of a Return Value in a Specific Error Code



**General Error Codes**

Table 2-2 explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a place holder and represents the number of the system function parameter that caused the error.

Table 2-2 General Error Codes

Error Code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x. This error was not caused by the user and cannot be eliminated by the user.
8x22 8x23	Range length error when reading a parameter. Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address or that the length of a bit range is not a multiple of 8 with an ANY parameter.
8x24 8x25	Range error when reading a parameter. Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x24 8x25	Range error when reading a parameter. Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer number that is too high. This error code indicates that the timer specified in parameter x does not exist.
8x27	The parameter contains a counter number that is too high (counter number error). This error code indicates that the counter specified in parameter x does not exist.
8x28 8x29	Alignment error when reading a parameter. Alignment error when writing a parameter. This error code indicates that the reference to parameter x is a bit address that is not equal to 0.
8x30 8x31	The parameter is located in a read-only global DB. The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32 8x34 8x35	The parameter contains a DB number that is too high (DB number error). The parameter contains an FC number that is too high (FC number error). The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A 8x3C 8x3E	The parameter contains the number of a DB that is not loaded. The parameter contains the number of an FC that is not loaded. The parameter contains the number of an FB that is not loaded.
8x42 8x43	An access error occurred while the system was attempting to read a parameter from the peripheral input area. An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44 8x45	Error in the nth ( $n > 1$ ) read access after an error occurred. Error in the nth ( $n > 1$ ) write access after an error occurred. This error code indicates that access to the required parameter is denied.

## 2.2 Meaning of the Parameters REQ, RET\_VAL and BUSY with Asynchronous SFCs

### Asynchronous SFCs

SFCs that operate asynchronously are SFCs that are called more than once before they complete their functions.

The following SFCs are either always executed asynchronously or in certain situations:

- SFC 7 “DP\_PRAL”
- SFC13 “DPNRM\_DG”
- SFC51 “RDSYSST”
- SFC55 “WR\_PARM”
- SFC56 “WR\_DPARM”
- SFC57 “PARM\_MOD”
- SFC58 “WR\_REC”
- SFC59 “RD\_REC”
- SFC65 “X\_SEND”
- SFC67 “X\_GET”
- SFC68 “X\_PUT”
- SFC69 “X\_ABORT”
- SFC72 “I\_GET”
- SFC73 “I\_PUT”
- SFC74 “I\_ABORT”

### Identifying the Job

If you trigger a hardware interrupt, start a data transfer, or abort a non-configured connection with one of the SFCs listed above and then call the same SFC again before the current function is completed, the reaction of the SFC will depend on whether or not the second call involves the same job.

Table 2-3 explains which input parameters specify the job for each of these SFCs. If these parameters match those of a job that is not yet completed, the SFC call counts as a follow-on call.

Table 2-3 Input Parameters for Job Identification

SFC	Job is identified by ...
7 "DP_PRAL"	IOID, LADDR
13 "DPNRM_DG"	LADDR
51 "RDSYSST"	SZL_ID, INDEX
55 "WR_PARM"	IOID, LADDR, RECNUM
56 "WR_DPARM"	IOID, LADDR, RECNUM
57 "PARM_MOD"	IOID, LADDR
58 "WR_REC"	IOID, LADDR, RECNUM
59 "RD_REC"	IOID, LADDR, RECNUM
65 "X_SEND"	DEST_ID, REQ_ID
67 "X_GET"	DEST_ID, VAR_ADDR
68 "X_PUT"	DEST_ID, VAR_ADDR
69 "X_ABORT"	DEST_ID
72 "I_GET"	IOID, LADDR, VAR_ADDR
73 "I_PUT"	IOID, LADDR, VAR_ADDR
74 "I_ABORT"	IOID, LADDR

### Input Parameter REQ

The REQ (request) input parameter is used solely to start the function:

- If you call the SFC for a job that is not currently active, the job is started by REQ = 1 (situation 1).
- If a particular job has been started and not yet completed and you call the SFC again to perform the same job (for example in a cyclic interrupt OB), then REQ is not evaluated by the SFC (situation 2).

**Output Parameters  
RET\_VAL and  
BUSY**

The status of the job is indicated by the output parameters RET\_VAL and BUSY.

- In situation 1 (first call with REQ=1), W#16#7001 is entered in RET\_VAL if system resources are free and the input parameters are correct. BUSY is then set.  
If the required system resources are currently being used or the input parameters have errors, the corresponding error code is entered in RET\_VAL and BUSY has the value 0. (Refer also to the note in Section 2.1)
- In situation 2 (call while the same job is active), W#16#7002 is entered in RET\_VAL (this is a warning that the job is still being processed), and BUSY is set.
- The following applies to the last call for a job:
  - With SFC 13 “DPNRM\_DG”, SFC67 “X\_GET” and SFC72 “I\_GET” the number of supplied data is entered in RET\_VAL as a positive number of bytes if no error occurred. BUSY then has the value 0. If an error occurs, RET\_VAL contains the error information and BUSY has the value 0. (Refer also to the note in Section 2.1)
  - With SFC59 “RD\_REC” the size of the data record in bytes is entered in RET\_VAL or the value 0 if no error occurred (see Section 7.6). In this case, BUSY has the value 0. If an error occurs, the error code is entered in RET\_VAL and BUSY has the value 0. (Refer also to the note in Section 2.1)
  - With all other SFCs, if the job was executed error-free, 0 is entered in RET\_VAL, and BUSY has the value 0. If an error occurs, the error code is entered in RET\_VAL and BUSY has the value 0. (Refer also to the note in Section 2.1)

---

**Note**

If the first and last call come together, the reaction is the same as described for the last call.

---

**Overview**

Table 2-4 provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the transfer of the job is not completed after an SFC has been called.

**Note**

Following every call, you must evaluate the relevant output parameters in your program.

Table 2-4 Relationship between the Call, REQ, RET\_VAL and BUSY with a Long Job

Number of the Call	Type of Call	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code	0
2 to (n - 1)	Intermediate call	Irrelevant	W#16#7002	1
n	Last call	Irrelevant	W#16#0000 (exceptions: SFC59 "RD_REC" if the destination area is larger than the data record transferred and SFC13 "DPNRM_DG", SFC67 "X_GET" and SFC72 "I_GET"), if no error has occurred	0
			Error code if errors occurred	0



# Copy and Block Functions

# 3

## Chapter Overview

Section	Description	Page
3.1	Copying Variables with SFC20 “BLKMOV”	3-2
3.2	Initializing a Memory Area with SFC21 “FILL”	3-4
3.3	Creating a Data Block with SFC22 “CREAT_DB”	3-6
3.4	Deleting a Data Block with SFC23 “DEL_DB”	3-8
3.5	Testing a Data Block with SFC24 “TEST_DB”	3-10
3.6	Compressing the User Memory with SFC25 “COMPRESS”	3-11
3.7	Transferring a Substitute Value to Accumulator 1 with SFC44 “REPL_VAL”	3-13

### 3.1 Copying Variables with SFC20 “BLKMOV”

#### Description

You use SFC20 “BLKMOV” (block move) to copy the content of a memory area (= source area) to another memory area (= destination area).

Using SFC20 “BLKMOV” you can copy all memory areas except the following:

- Blocks: FB, SFB, FC, SFC, OB, SDB,
- Counters,
- Timers,
- Memory areas of the peripheral I/O areas.

The source parameter can be a part of the data block in the load memory which is not relevant to program execution (DB compiled with the keyword UNLINKED).

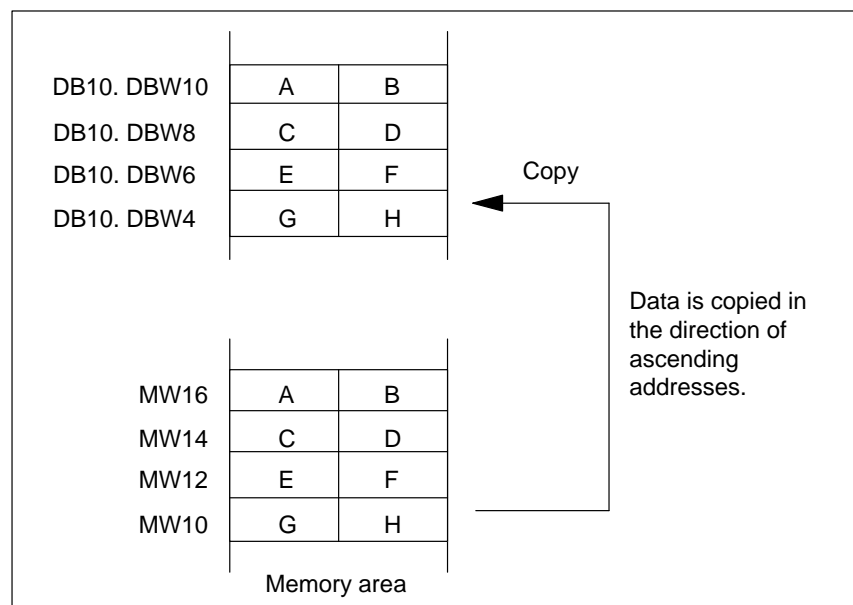


Figure 3-1 Copying Memory Contents with SFC20 “BLKMOV”

#### Interruptability

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, SFC20 is interrupted while copying from a DB that is not relevant to program execution, the execution of SFC20 can no longer be nested.



## Parameters

Table 3-1 Parameters for SFC20 “BLKMOV”

Parameter	Declaration	Data Type	Memory Area	Description
SRCBLK	INPUT	ANY	I, Q, M, D, L	Specifies the memory area to be copied (source area).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs when the function is being executed, the return value contains an error code.
DSTBLK	OUTPUT	ANY	I, Q, M, D, L	Specifies the memory area to which the data will be copied (destination area).

---

### Note

The source and destination areas must not overlap. If the specified destination area is larger than the source area, the function only copies as much data to the destination area as is contained in the source area.

If the specified destination area is smaller than the source area, the function only copies as much data as can be written to the destination area.

---

## Error Information

How you evaluate the error information of the parameter RET\_VAL is explained in Chapter 2. This chapter also contains the general error information of the SFCs. SFC20 does not output any specific error information.

### 3.2 Initializing a Memory Area with SFC21 “FILL”

#### Description

With SFC21 “FILL”, you can initialize a memory area (destination area) with the content of another memory area (source area). The SFC copies the content of the specified destination area until the memory area is completely full. The defined variable is written to the memory area in the ascending order of addresses.

#### Note

If the destination area to be initialized is not a whole multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be initialized is smaller than the source area, the function only copies as much data as can be written to the destination area.

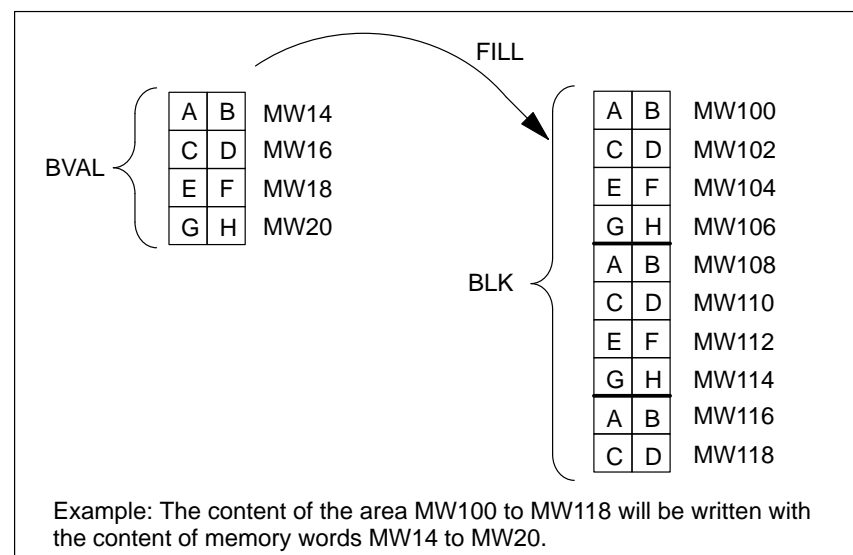


Figure 3-2 Principle of Initializing a Memory Area

#### Exceptions

You cannot write values to the following using SFC21:

- Blocks: FB, SFB, FC, SFC, SDB,
- Counters,
- Timers,
- Memory areas of the peripheral I/O area.

## Parameters

Table 3-2 Parameters for SFC21 "FILL"

Parameter	Declaration	Data Type	Memory Area	Description
BVAL	INPUT	ANY	I, Q, M, D, L	The parameter BVAL contains the value or description of the area whose contents will be used to initialize the destination area (source area).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being processed, the return value contains an error code.
BLK	OUTPUT	ANY	I, Q, M, D, L	The parameter BLK contains the description of the area to be initialized (destination area).

### The Input Parameter is a Structure

If you transfer a structure as the input parameter, remember the following point:

STEP 7 always defines the length of a structure as an even number of bytes. As a result, the structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

#### Example

The structure was declared as follows:

```
TYP_5_BYTE_STRUCTURE:STRUCT
  BYTE_1_2: WORD
  BYTE_3_4: WORD
  BYTE_5: BYTE
END_STRUCT
```

The declared structure "TYP\_5\_BYTE\_STRUCTURE" requires 6 bytes of memory.

### Error Information

How you evaluate the error information of the parameter RET\_VAL is explained in Chapter 2. This chapter also contains the general error information of the SFCs. SFC21 does not output specific error information with the RET\_VAL parameter.

### 3.3 Creating a Data Block with SFC22 “CREAT\_DB”

#### Description

With SFC22 “CREAT\_DB” (create data block), you create a data block that does not contain initialized values. The SFC creates a data block of a selectable length with a block number taken from a specified range. The SFC assigns the lowest possible number to the DB from the specified range. If you want to create a DB with a particular number, simply select the range specifying the same value as the upper and lower limit. You cannot assign a number if a DB with the same number already exists in the user program. The length of the DB must be an even number of bytes.

#### Interruptability

SFC22 “CREAT\_DB” can be interrupted by higher priority OBs. If SFC22 “CREAT\_DB” is called again in a higher priority OB, remember the following points

- The number that the interrupted SFC22 will assign to the data block when it resumes execution is no longer available.
- The nesting depth depends on the particular CPU.

#### Parameters

Table 3-3 Parameters for SFC22 “CREAT\_DB”

Parameter	Declaration	Data Type	Memory Area	Description
LOW_LIMIT	INPUT	WORD	I, Q, M, D, L, constant	The lower limit value is the smallest number in the range of numbers that you can assign to your data block.
UP_LIMIT	INPUT	WORD	I, Q, M, D, L, constant	The upper limit value is the highest number in the range of numbers you can assign to your data block.
COUNT	INPUT	WORD	I, Q, M, D, L, constant	The count value specifies the number of data bytes you want to reserve for your data block. Here you must specify an even number of bytes (maximum 65534).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.
DB_NUMBER	OUTPUT	WORD	I, Q, M, D, L	The data block number is the number of the created data block.

## Error Information

Table 3-4 Specific Error Information for SFC22 "CREAT\_DB"

Error Code (W#16#...)	Explanation
0000	No error occurred.
8091	The nesting level was exceeded.
8092	The compress function is currently active.
80A1	Error in the number of the DB: <ul style="list-style-type: none"> <li>The number is 0.</li> <li>The number exceeds the number of DBs for the specific CPU.</li> <li>Parameter lower limit &gt; upper limit.</li> </ul>
80A2	Error in the length of the DB: <ul style="list-style-type: none"> <li>The length is 0.</li> <li>The length was specified as an odd number.</li> <li>The length is greater than permitted by the CPU.</li> </ul>
80B1	There is no DB number free.
80B2	There is not enough free memory available.
80B3	There is not enough continuous memory space available (remedy: compress memory!)

### 3.4 Deleting a Data Block with SFC23 “DEL\_DB”

#### Description

With SFC23 “DEL\_DB” (delete data block) you delete a data block located in the work memory and, if present, in the load memory of the CPU. The DB to be deleted must not be open in the current or in any lower priority class, in other words, it must not be entered in either of the two DB registers or in the B stack. Otherwise the CPU switches to the STOP mode when SFC23 is called.

Table 3-5 explains when a DB can be deleted with SFC23 “DEL-DB”.

Table 3-5 Relationship between the creation of a DB and its erasability with SFC23

If ...	Then ...
The DB was created by calling SFC22 “CREAT_DB”,	SFC23 can delete it.
The DB was transferred to the CPU by STEP 7 and was not created with the keyword UNLINKED,	SFC23 can delete it.
The DB is located on a flash card,	SFC23 cannot delete it.

#### Interruptability

SFC23 “DEL\_DB” can be interrupted by priority classes of a higher priority. If the SFC is again called there, then this second call is aborted and W#16#8091 is entered in RET\_VAL.

#### Parameters

Table 3-6 Parameters for SFC23 “DEL\_DB”

Parameter	Declaration	Data Type	Memory Area	Description
DB_NUMBER	INPUT	WORD	I, Q, M, D, L, constant	Number of the DB to be deleted
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information

**Error Information**

Table 3-7 Specific Error Information for SFC23 “DEL\_DB”

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8091	SFC23 calls were nested and the maximum nesting level of the CPU used was exceeded.
8092	The function “delete DB” cannot currently be executed since <ul style="list-style-type: none"><li>• The “compress user memory” function is currently active</li><li>• The “save user program” function is currently active.</li></ul>
80A1	Error in the input parameter DB_NUMBER: the actual parameter selected <ul style="list-style-type: none"><li>• is 0</li><li>• is greater than the maximum permitted DB number for the CPU used.</li></ul>
80B1	The DB with the specified number does not exist on the CPU.
80B2	The DB with the specified number created using the keyword UNLINKED.
80B3	The DB is on a flash card.

### 3.5 Testing a Data Block with SFC24 “TEST\_DB”

**Description** With SFC24 “TEST\_DB” (test data block), you obtain information about a data block located in the work memory of the CPU. The SFC queries the number of data bytes in the selected DB and checks whether or not the DB is read only.

#### Parameters

Table 3-8 Parameters for SFC24 “TEST\_DB”

Parameter	Declaration	Data Type	Memory Area	Description
DB_NUMBER	INPUT	WORD	I, Q, M, D, L, constant	Number of the DB to be tested
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
DB_LENGTH	OUTPUT	WORD	I, Q, M, D, L	Number of data bytes the selected DB contains.
WRITE_PROT	OUTPUT	BOOL	I, Q, M, D, L	Information about the write-protect identifier of the DB (1 means read only).

#### Error Information

Table 3-9 Specific Error Information for SFC24 “TEST\_DB”

Error Code (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in the input parameter DB_NUMBER: the actual parameter selected <ul style="list-style-type: none"> <li>is 0</li> <li>is greater than the max. permissible DB number for the CPU used.</li> </ul>
80B1	The DB with the specified number does not exist on the CPU.
80B2	The DB was created using the keyword UNLINKED.



### 3.6 Compressing the User Memory with SFC25 “COMPRESS”

#### Gaps in Memory

Gaps can occur in the load memory and in the work memory if data blocks are deleted and reloaded several times. These gaps reduce the effective memory area.

#### Description

With SFC25 “COMPRESS”, you start compression of the RAM section of both the load memory and the work memory. The compression function is the same as when started externally in the RUN-P mode (mode selector setting).

If compression was started externally and is still active, the SFC25 call will result in an error message.

---

#### Note

Data blocks with a length greater than the maximum permitted length for the particular CPU (check SZL-ID “#16#0131 INDEX W#16#0004”) are not shifted with SFC25 “COMPRESS”. This means that gaps may still remain in the work memory after compression.

---

#### Parameters

Table 3-10 Parameters for SFC25 “COMPRESS”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Indicates whether the compression function started by an SFC25 call is still active (1 means active).
DONE	OUTPUT	BOOL	I, Q, M, D, L	Indicates whether the compression function started by SFC25 was completed successfully (1 means completed successfully).

**Checking the  
Compression  
Function**

If SFC25 “COMPRESS” is called once the compression function is started. You cannot, however, check whether the memory was successfully compressed.

If you want to check the compression function, follow the steps outlined below:

Call SFC25 cyclically. First evaluate the parameter RET\_VAL after every call. Provided that its value is 0, the parameters BUSY and DONE can be evaluated. If BUSY = 1 and DONE = 0, this indicates that the compression function is still active. When BUSY changes to value 0 and DONE to the value 1, this indicates that the compression function was completed successfully. If SFC25 is called again afterwards, the compression function is started again.

**Error Information**

Table 3-11 Specific Error Information for SFC25 “COMPRESS”

Error Code (W#16#...)	Explanation
0000	No error occurred. The compression function was started by SFC25. Evaluation of the output parameters BUSY and DONE by the user program (see above) only provides useful information when this is the case.
8091	The compression function was started externally and is still active.
8092	The “compress user memory” function cannot currently be executed since <ul style="list-style-type: none"><li>the “delete data block” function was started externally and is still active</li><li>a test and start-up function currently requires a particular block (for example, status)</li><li>the “copy blocks” function was triggered externally and is still active.</li></ul>

### 3.7 Transferring a Substitute Value to Accumulator 1 with SFC44 “REPL\_VAL”

<b>Description</b>	With SFC44 “REPL_VAL” (replace value), you transfer a value to accumulator 1 of the priority class that caused the error.
<b>Restriction: only in Synchronous Error OBs</b>	You can only call SFC44 “REPL_VAL” in a synchronous error OB (OB121, OB122).
<b>Example of an Application</b>	If an input module is damaged to such an extent that no more values can be read from it, then each time the module is accessed, OB122 is started. Using SFC44, a suitable value in OB122 can be transferred to accumulator 1 of the interrupted priority class so that the program can continue with this substitute value. The information for selecting the substitute value (for example, the block in which the error occurred or the address affected) is located in the local variables of OB122.

#### Parameters

Table 3-12 Parameters for SFC44 “REPL\_VAL”

Parameter	Declaration	Data Type	Memory Area	Description
VAL	INPUT	DWORD	I, Q, M, D, L, constant	Substitute value
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.

#### Error Information

Table 3-13 Specific Error Information for SFC44 “REPL\_VAL”

Error Code (W#16#...)	Explanation
0000	No error occurred. A substitute value was entered.
8080	SFC44 was not called by a synchronous error OB (OB121, OB122).



# SFCs for Controlling Program Execution

# 4

## Chapter Overview

Section	Description	Page
4.1	Retriggering Cycle Time Monitoring with SFC43 "RE_TRIGR"	4-2
4.2	Changing the CPU to STOP with SFC46 "STP"	4-3
4.3	Delaying Execution of the User Program with SFC47 "WAIT"	4-4
4.4	Triggering a Multicomputing Interrupt with SFC35 "MP_ALM"	4-5

## 4.1 Retriggering Cycle Time Monitoring with SFC43 "RE\_TRIGR"

<b>Description</b>	With SFC43 "RE_TRIGR" (retrigger watchdog) you can retrigger the cycle time monitoring.
<b>Parameters</b>	SFC43 "RE_TRIGR" has no parameters.
<b>Error Information</b>	SFC43 "RE_TRIGR" does not provide any error information.

## **4.2 Changing the CPU to STOP with SFC46 “STP”**

**Description** With SFC46 “STP” (stop), you change the CPU to the STOP mode.

**Parameters** SFC46 “STP” has no parameters.

**Error Information** SFC46 “STP” does not provide any error information.

### 4.3 Delaying Execution of the User Program with SFC47 “WAIT”

**Description** With SFC47 “WAIT”, you program delays or waiting times in your user program. You can program waiting times up to 32767  $\mu$ s. The smallest possible waiting time depends on the particular CPU and is the same as the execution time of SFC47.

**Interruptability** SFC47 “WAIT” can be interrupted by higher priority OBs.

---

**Note**  
**(not for S7-400)**

The waiting time programmed with SFC47 is extended by the execution time of the nested priority classes.

---

#### Parameters

Table 4-1 Parameter for SFC47 “WAIT”

Parameter	Declaration	Data Type	Memory Area	Description
WT	INPUT	INT	I, Q, M, D, L, constant	The parameter WT contains the waiting time in $\mu$ s.

**Error information** SFC47 “WAIT” does not provide any error information.



## 4.4 Triggering a Multicomputing Interrupt With SFC 35 "MP\_ALM"

### Description

Calling SFC 35 "MP\_ALM" during multicomputing triggers the multicomputing interrupt. This leads to a synchronized start of OB60 on all CPUs involved. In the single processor mode and when operating with a segmented rack, OB60 is only started on the CPU that called SFC35.

You can indicate the cause of the multicomputing interrupt using the JOB input parameter. This job identifier is transferred to all the CPUs involved and you can evaluate it in OB60 (see Section 1.7 and /234/).

You can call SFC 35 "MP\_ALM" at any point in your program. Since the call would be pointless in any mode other than RUN, if it is called in the STARTUP mode, the multicomputing interrupt is suppressed. The function value informs you of this.

### Parameters

Table 4-2 Parameters for SFC 35 "MP\_ALM"

Parameter	Declaration	Data Type	Memory Area	Description
JOB	INPUT	BYTE	I, Q, M, D, L, const.	Job identifier Possible values: 1 to 15
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs during execution of the function, the return value contains an error code.

### Error Information

Table 4-3 Error Information Specific to SFC 35 "MP\_ALM"

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	The JOB input parameter contains an illegal value.
80A0	Execution of OB60 following the last multicomputing interrupt is not completed either on the local or on another CPU.
80A1	Incorrect mode (STARTUP instead of RUN).



# SFCs for Handling the System Clock

# 5

## Chapter Overview

Section	Description	Page
5.1	Setting the Time with SFC0 “SET_CLK”	5-2
5.2	Reading the Time with SFC1 “READ_CLK”	5-3
5.3	Synchronizing Slave Clocks with SFC48 “SNC_RTCB”	5-4

## 5.1 Setting the Time with SFC0 “SET\_CLK”

### Description

With SFC0 “SET\_CLK” (set system clock), you set the time and the date of the CPU clock. The SFC0 call starts the clock. The clock then runs starting from the set time and set date.

If the clock is a master clock, the CPU also starts to synchronize the time when SFC0 is called. You set the synchronization intervals using STEP 7.

### Parameters

Table 5-1 Parameters for SFC0 “SET\_CLK”

Parameter	Declaration	Data Type	Memory Area	Description
PDT	INPUT	DT	D, L, constant	At the PDT input, you enter the date and time you want to set.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs during the execution of the function, the return value contains an error code.

### Date and Time

You enter the date and time as data type DT. As an example: for January 15th, 1995, 10:30 a.m. and 30 seconds you would enter: DT#1995-01-15-10:30:30. The time can only be entered with a granularity of seconds. The day of the week is calculated by SFC0 “SET\_CLK” from the date.

Remember that you must first create the data type DT with the FC “D\_TOD\_DT” before you can transfer input parameters to it (see Section 21.4).

### Error Information

Table 5-2 Specific Error Information for SFC0 “SET\_CLK”

Error Code (W#16#...)	Explanation
0000	No error
8080	Error in date
8081	Error in time

## 5.2 Reading the Time with SFC1 “READ\_CLK”

**Description** With SFC1 “READ\_CLK” (read system clock), you read the current date or current time of the system clock of the CPU.

### Parameters

Table 5-3 Parameters for SFC1 “READ\_CLK”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs during the execution of the function, the return value contains an error code.
CDT	OUTPUT	DT	D,L	The current date and current time are output at the CDT output.

**Error Information** How you evaluate the error information of the RET\_VAL parameter is explained in Chapter 2. This chapter also contains the general error information for SFCs. SFC1 does not output any specific error information.

### 5.3 Synchronizing Slave Clocks with SFC48 “SNC\_RTCB”

**Definition:** Synchronizing slave clocks refers to the transmission of the date and time from the master clock of a bus segment (for example, the S7-400 K-bus, MPI, or S7 backplane bus) to all clock slaves of the bus segment.

**Description** With SFC48 “SNC\_RTCB” (synchronize real time clocks) you synchronize all the slave clocks on a bus segment. Successful synchronization is only possible when SFC48 is called on a CPU whose real-time clock was assigned the master clock function for at least one bus segment. You assign the relevant parameters with STEP 7.

The system synchronization of the slave clocks (cyclic after the selected synchronization interval has elapsed) is independent of SFC48 calls.

#### Parameters

Table 5-4 Parameters for SFC48 1 “SNC\_RTCB”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs during the execution of the function, the return value contains an error code.

#### Error Information

Table 5-5 Specific Error Information for SFC48 ”SNC\_RTCB”

Error Code (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.

# SFCs for Handling Run-Time Meters

# 6

Chapter Overview

Section	Description	Page
6.1	Run-Time-Meters	6-2
6.2	Setting the Run-Time Meter with SFC2 “SET_RTM”	6-3
6.3	Starting and Stopping a Run-Time Meter with SFC3 “CTRL_RTM”	6-4
6.4	Reading a Run-Time Meter with SFC4 “READ_RTM”	6-5
6.5	Reading the System Time with SFC64 “TIME_TCK”	6-6

## 6.1 Run-Time Meters

<b>Introduction</b>	The CPUs have a number of run-time meters (refer to the data sheets of your CPU). Using SFCs 2, 3 and 4, you can set, stop and read run-time meters.
<b>Application</b>	<p>You can use a run-time meter for a variety of applications:</p> <ul style="list-style-type: none"><li>• For measuring the run-time of the CPU</li><li>• For measuring the run-time of controlled apparatus or connected devices</li></ul>
<b>Characteristics of the Run-Time Meter</b>	<p>When it is started, the run-time meter begins to count starting at the last recorded value. If you want it to start at a different initial value, you must set this value with SFC2. If the CPU changes to the STOP mode, or you stop the run-time meter, the CPU records the current value of the run-time meter. When the CPU is started again (complete restart), the run-time meter must be started again with SFC3.</p>
<b>Range of Values</b>	Each run-time meter has a range of values from 0 to 32 767 hours.



## 6.2 Setting the Run-Time Meter with SFC2 “SET\_RTM”

**Description** With SFC2 “SET\_RTM” (set run-time meter), you set a run-time meter of the CPU to a selected value. The number of run-time meters you can set depends on the particular CPU you are using.

### Parameters

Table 6-1 Parameters for SFC2 “SET\_RTM”

Parameter	Declaration	Data Type	Memory Area	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input NR contains the number of the run-time meter you want to set (possible values: 0 to 7).
PV	INPUT	INT	I, Q, M, D, L, constant	Input PV contains the setting for the run-time meter (default).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.

### Error Information

Table 6-2 Specific Error Information for SFC2 “SET\_RTM”

Error Code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the run-time meter
8081	A negative value was transferred to the PV parameter.

### 6.3 Starting and Stopping a Run-Time Meter with SFC3 “CTRL\_RTM”

**Description** With SFC3 “CTRL\_RTM” (control run-time meter), you can start or stop a run-time meter of the CPU.

#### Parameters

Table 6-3 Parameters for SFC3 “CTRL\_RTM”

Parameter	Declaration	Data Type	Memory Area	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input NR contains the number of the run-time meter you want to start or stop (possible values: 0 to 7).
S	INPUT	BOOL	I, Q, M, D, L, constant	Input S starts or stops the run-time meter. Set the signal state to “0” when you want to stop the counter. Set the signal state to “1” when you want to start the counter.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.

#### Error Information

Table 6-4 Specific Error Information for SFC3 “CTRL\_RTM”

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the run-time meter

## 6.4 Reading a Run-Time Meter with SFC4 “READ\_RTM”

### Description

With SFC4 “READ\_RTM” (read run-time meter), you read a run-time meter. SFC4 provides the current run time as output data and the status of the counter, for example “stopped” or “counting”.

If the run-time meter runs for longer than 32767 hours, it stops at the count 32767 and outputs the error message “overflow”.

### Parameters

Table 6-5 Parameters for SFC4 “READ\_RTM”

Parameter	Declaration	Data Type	Memory Area	Description
NR	INPUT	BYTE	I, Q, M, D, L, constant	Input NR contains the number of the run-time meter you want to read (possible values: 0 to 7).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while processing the function, the return value contains an error code.
CQ	OUTPUT	BOOL	I, Q, M, D, L	Output CQ indicates whether the run-time meter is running or stopped. The signal state “0” shows that the run-time meter is stopped. Signal state “1” shows that the run-time meter is running.
CV	OUTPUT	INT	I, Q, M, D, L	Output CV indicates the current value of the run-time meter.

### Error Information

Table 6-6 Specific Error Information for SFC4 “READ\_RTM”

Error Code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the run-time meter
8081	Overflow of the run-time meter

## 6.5 Reading the System Time with SFC64 “TIME\_TCK”

**Description** With SFC64 “TIME\_TCK” (time tick), you can read the system time of the CPU. The system time is a “time counter” counting cyclically from 0 to a maximum of 2147483647 ms. In case of an overflow the system time is counted again starting with 0. The resolution and the accuracy of the system time are 10 ms for the S7-300 and 1 ms for the S7-400. The system time is influenced only by the operating modes of the CPU.

**Application** You can use the system time for example to measure the duration of processes by comparing the results of two SFC64 calls.

### System Time and Modes

Table 6-7 System Time Dependent on the Modes

Mode	System time ...
Startup	... is constantly updated
RUN	
STOP	... is stopped and retains the current value
Restart (not with S7-300)	... continues with the value saved at the change to the STOP mode
Complete restart	... is deleted and restarts with “0”

### Parameters

Table 6-8 Parameter for SFC64 “TIME\_TCK”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to $2^{31}-1$ ms.

**Error Information** SFC64 “TIME\_TCK” does not provide any error information.

# SFCs for Transferring Data Records

# 7

## Chapter Overview

Section	Description	Page
7.1	Writing and Reading Data Records	7-2
7.2	Writing Dynamic Parameters with SFC55 “WR_PARM”	7-4
7.3	Writing Default Parameters with SFC56 “WR_DPARM”	7-5
7.4	Assigning Parameters to a Module with SFC57 “PARM_MOD”	7-6
7.5	Writing a Data Record with SFC58 “WR_REC”	7-8
7.6	Reading a Data Record with SFC59 “RD_REC”	7-9
7.7	Reading a Data Record with SFC59 “RD_REC” on S7-300 CPUs	7-13
7.8	Further Error Information from SFCs 55 to 59	7-16

## 7.1 Writing and Reading Data Records

### Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see Table 7-1).

Other modules have a read-only system data area in which your program can read data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see Table 7-2).

### Note

There are modules that have both system data areas. These are physically separate areas and all they have in common is their logical structure.

### Write-Only System Data Area

Table 7-1 shows the structure of the write-only system data area. This table also shows how long the data records can be and with which SFCs the data records can be written.

Table 7-1 Structure of the Write-Only System Data Area on Modules

Data Record Number	Content	Size	Restriction	Can be written with SFC
0	Parameters	With S7-300: from 2 to 14 bytes	Can only be written by an S7-400	56 "WR_DPARM" 57 "PARM_MOD"
1	Parameters	With S7-300: from 2 to 14 bytes Data records 0 and 1 together have a total of exactly 16 bytes.	–	55 "WR_PARM" 56 "WR_DPARM" 57 "PARM_MOD"
2 to 127	User data	Each ≤ 240 bytes	–	55 "WR_PARM" 56 "WR_DPARM" 57 "PARM_MOD" 58 "WR_REC"
128 to 240	Parameters	Each ≤ 240 bytes	–	55 "WR_PARM" 56 "WR_DPARM" 57 "PARM_MOD" 58 "WR_REC"

### Read-Only System Data Area

Table 7-2 shows the structure of the read-only system data area. This table also shows how long the data records can be and with which SFCs the data records can be read.

Table 7-2 Structure of the Read-Only System Data Area on Modules

Data Record Number	Content	Size	Can be read with SFC
0	Module-specific diagnostic data	4 bytes	51 “RDSYSST” (INDEX 00B1H) 59 “RD_REC”
1	Channel-specific diagnostic data (including data record 0)	<ul style="list-style-type: none"> <li>• With S7-300: 16 bytes</li> <li>• With S7-400: from 7 to 220 bytes</li> </ul>	51 “RDSYSST” (INDEX 00B2H and 00B3H) 59 “RD_REC”
2 to 127	User data	Each ≤ 240 bytes	59 “RD_REC”
128 to 240	Diagnostic data	Each ≤ 240 bytes	59 “RD_REC”

**System Resources**

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If all the available system resources are being used, this is indicated in RET\_VAL. You can remedy this temporary error situation by simply repeating the job.

The maximum number of “simultaneously” active SFC jobs depends on the CPU. Refer to **/70/** and **/101/** for more detailed information.

## 7.2 Writing Dynamic Parameters with SFC55 “WR\_PARM”

**Description** With SFC55 “WR\_PARM” (write parameter), you transfer the data record RECORD to the addressed module. The parameters transferred to the module do **not** overwrite the parameters of this module in the corresponding SDB if they exist there.

**Prior Conditions** The data record to be transferred must not be static:

- It must not be data record 0 (data record 0 is static throughout the system).
- If the data record is referenced in SDBs 100 to 129, the static bit must not be set.

Refer to /71/ and /101/ for more information about static data records.

### Parameters

Table 7-3 Parameters for SFC55 “WR\_PARM”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 =Peripheral input (PI) B#16#55 =Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Data record number
RECORD	INPUT	ANY	I, Q, M, D, L	Data record
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Writing is not yet completed.

**Input Parameter RECORD** The data to be transferred are read from the parameter RECORD during the first SFC call. If the transfer of the data record takes longer than the duration of a call, the content of the parameter RECORD is no longer relevant for the subsequent SFC calls (for the same job).

**Error Information** See Table 7-6.

#### Note (only for S7-400)

If the general error W#16#8544 occurs, this only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.



### 7.3 Writing Default Parameters with SFC56 “WR\_DPARM”

#### Description

With SFC56 “WR\_DPARM” (write default parameter), you transfer the data record with the number RECNUM from the corresponding SDB (SDB100 to SDB103 for S7-300, SDB100 to SDB129 for S7-400) to the addressed module. With this function, it is irrelevant whether the data record is static or dynamic.

#### Parameters

Table 7-4 Parameters for SFC56 “WR\_DPARM”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Data record number
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Writing is not yet completed.

#### Error Information

See Table 7-6.

## 7.4 Assigning Parameters to a Module with SFC57 “PARM\_MOD”

### Description

With SFC57 “PARM\_MOD” (assign parameters to a module) you transfer all the data records of a module that you configured with STEP 7 in the corresponding SDB (SDB100 to SDB103 for S7-300, SDB100 to SDB129 for S7-400) to the module. With this function, it is irrelevant whether the data records are static or dynamic.

### Parameters

Table 7-5 Parameters for SFC57 “PARM\_MOD”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Writing is not yet completed.

### Error Information

The “real” error information (error codes W#16#8xyz) in Table 7-6 can be divided into two classes:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):  
With this type of error, it is possible that the error will be eliminated without you taking any action, in other words, it is advisable to call the SFC again (if necessary more than once).  
An example of a temporary error is when required resources are currently being used (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A1, 80Bx, 80Dx):  
This type of error will not be eliminated without you taking action. Calling the SFC again will only be successful after the error has been eliminated.  
An example of a permanent error is entering the wrong length in RECORD (W#16#80B1).

Table 7-6 Specific Error Information for SFC55 “WR\_PARM”, SFC56 “WR\_DPARM” and SFC57 “PARM\_MOD”

Error Code (W#16#...)	Explanation	Restriction
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	–
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/Os
7002	Interim call (REQ irrelevant): Data transfer active; BUSY has the value 1.	Distributed I/Os
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	–
8092	The type specified in the ANY reference is not BYTE.	only with S7-400 for SFC55 (WR_PARM)
8093	This SFC is not permitted for the module specified by LADDR and IOID (the following modules are permitted: S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	–
80A1	Negative acknowledgement when sending the data record to the module (the module was removed or became defective during transfer).	–
80A2	DP protocol error at layer 2, possibly hardware fault.	Distributed I/Os
80A3	DP protocol error with direct data link mapper or user interface/user. Possibly hardware fault.	Distributed I/Os
80A4	Communication problem on communication bus	Error occurs between the CPU and external DP interface module
80B0	SFC for module type not possible, module does not recognize the data record.	–
80B1	The length of the transferred data record is incorrect.	–
80B2	The configured slot is not occupied.	–
80B3	Actual module type does not match the required module type in SDB1	–
80C1	The data of the previous write job for the same data record on the module have not yet been processed by the module.	–
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	–
80C3	The required resources (memory etc.) are currently occupied.	–
80C4	Communication errors: <ul style="list-style-type: none"> <li>• Parity error</li> <li>• Software ready not set</li> <li>• Error in field length information</li> <li>• Checksum error on the CPU side</li> <li>• Checksum error on the module side</li> </ul>	–
80C5	Distributed I/Os not available.	Distributed I/Os
80C6	Data record transfer was stopped due to a priority class abort (restart or background)	Distributed I/Os
80D0	There is no entry for the module in the corresponding SDB.	–
80D1	The data record number is not configured in the corresponding SDB for the module (data record numbers $\geq 241$ are rejected by STEP 7).	–
80D2	The module cannot be assigned parameters according to its type identifier.	–
80D3	The SDB cannot be accessed since it does not exist.	–
80D4	SDB structure error: The SDB internal pointer points to a value outside the SDB.	only with S7-300
80D5	The data record is static.	only with SFC55 (WR_PARM)

## 7.5 Writing a Data Record with SFC58 “WR\_REC”

### Description

With SFC58 “WR\_REC” (write record), you transfer the data record contained in RECORD to the addressed module.

You start the write job by assigning the value 1 to the input parameter REQ when SFC58 is called. If the write job could be executed immediately, the SFC returns the value 0 at the output parameter BUSY. If BUSY has the value 1, writing is not yet completed (see Section 2.2).

### Parameters

Table 7-7 Parameters for SFC58 “WR\_REC”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Write request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) With mixed modules, specify the area ID of the lowest address. With the same addresses, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Data record number (permitted values: 2 to 240)
RECORD	INPUT	ANY	I, Q, M, D, L	Data record. Only the data type BYTE is permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Writing is not yet completed.

### Input Parameter RECORD

The data to be transferred are read from the parameter RECORD during the first SFC call. If the transfer of the data record takes longer than the duration of a call, the content of the parameter RECORD is no longer relevant for the subsequent SFC calls (for the same job).

### Error Information

See Table 7-9

#### Note (only for S7-400)

If the general error W#16#8544 occurs, this only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

## 7.6 Reading a Data Record with SFC59 “RD\_REC”

### Description

With SFC59 “RD\_REC” (read record), you read the data record with the number RECNUM from the addressed module. You start the read job by calling SFC59 and assigning the value 1 to the input parameter REQ. If the read job could be executed immediately, the SFC returns the value 0 in the BUSY output parameter. If BUSY has the value 1, the read job is not yet completed (see Section 2.2). The data record read is entered in the destination area indicated by the RECORD parameter providing the data transfer was free of errors.

If you read a data record with a number higher than one from an FM or CP purchased before February 1997 (called an “older module” below), The reaction of SFC59 is different from that with a new module. This special situation is described in the section “Using older S7-300 FMs and CPs with data record numbers higher than 1”.

### Parameters

Table 7-8 Parameters for SFC58 “RD\_REC”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Read request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Data record number (permitted values 0 to 240)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code. The length of the data record actually transferred in bytes (possible values: +1 to +240) is also entered if the destination area is larger than the transferred data record and if no error occurred in the transfer.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Reading is not yet completed.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the data record read. With asynchronous execution of SFC59, make sure that the actual parameters of RECORD have the same length information for all calls. Only data type BYTE is permitted.

**Output Parameter  
RET\_VAL**

- If an error occurred while the function was being executed, the return value contains an error code.
- If no error occurred, RET\_VAL contains the following:
  - 0: if the entire destination area was filled with data from the selected data record (the data record can also be incomplete).
  - the length of the data record actually transferred in bytes (possible values: +1 to + 240) if the destination area is larger than the transferred data record.

---

**Note**

**(only for the S7-400)**

If the general error W#16#8545 occurs, this only indicates that access to at least one byte of the I/O memory area containing the data record was blocked. The data record was read by the module correctly and written to the I/O memory area.

---

**Setting RECORD**

---

**Note**

If you want to ensure that the entire data record is always read, select a destination area with a length of 241 bytes. If the data transfer is error-free, RET\_VAL contains the actual data record length.

---

**Using older S7-300  
FMs and CPs with  
Data Record  
Numbers higher  
than 1**

If you want to read out a data record with a number higher than 1 from an older S7-300 FM or older S7-300 CP using SFC59 “RD\_REC”, remember the following points:

- If the destination area is larger than the actual length of the required data record, no data are entered in RECORD.  
RET\_VAL has the value W#16#80B1.
- If the destination area is smaller than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes in RECORD.  
RET\_VAL has the value 0.
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD. RET\_VAL has the value 0.

**Error Information**

The “real” error information (error codes W#16#8xyz) in Table 7-9 can be divided into two classes:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):  
With this type of error, it is possible that the error will be eliminated without you taking any action, in other words, it is advisable to call the SFC again (if necessary more than once).  
An example of a temporary error is when required resources are currently being used (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A1, 80Bx, 80Dx):  
This type of error will not be eliminated without you taking action. Calling the SFC again will only be successful after the error has been eliminated.  
An example of a permanent error is entering the wrong length in RECORD (W#16#80B1).

Table 7-9 Specific Error Information for SFC58 “WR\_REC” and SFC59 “RD\_REC”

Error Code (W#16#...)	Explanation	Restriction
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	–
7001	First call with REQ=1: No data transfer active; BUSY has the value 1.	Distributed I/Os
7002	Interim call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/Os
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	–
8092	The type specified in the ANY reference is not BYTE.	only with S7-400
8093	This SFC is not permitted for the module specified by LADDR and IOID (the following modules are permitted: S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	–
80A0	Negative acknowledgement when reading from the module (the module was removed during the read job or is defective).	only with SFC59 (RD_REC)
80A1	Negative acknowledgement when sending the data record to the module (the module was removed during transfer or is defective).	only with SFC58 (WR_REC)
80A2	DP protocol error at layer 2, possibly hardware fault.	Distributed I/Os
80A3	DP protocol error with direct data link mapper or user interface/user. Possibly hardware fault.	Distributed I/Os
80A4	Communication problem on the K bus	The error occurs between the CPU and the external DP interface module.
80B0	<ul style="list-style-type: none"> <li>• SFC not possible for module type.</li> <li>• The module does not recognize the data record.</li> <li>• Data record number <math>\geq 241</math> not permitted.</li> <li>• With SFC58 (WR_REC), data records 0 and 1 are not permitted.</li> </ul>	–

Table 7-9 Specific Error Information for SFC58 “WR\_REC” and SFC59 “RD\_REC”, continued

Error Code (W#16#...)	Explanation	Restriction
80B1	The length specified in the RECORD parameter is incorrect.	<ul style="list-style-type: none"> <li>With SFC58 (WR_REC): Length incorrect</li> <li>With SFC59 (RD_REC) (only when using older S7-300 SMs and S7-300 CPs): <ul style="list-style-type: none"> <li>S7-300: length &gt; rec. length</li> <li>S7-400: length &lt; rec. length</li> </ul> </li> <li>With SFC13 (DPNRM_DG): length &lt; rec. length</li> </ul>
80B2	The configured slot is not occupied.	–
80B3	Actual module type does not match the required module type in SDB1	–
80C0	<ul style="list-style-type: none"> <li>With SFC59 (RD_REC): The module has the data record, but there are still no data to be read.</li> <li>With SFC13 (DPNRM_DG): There are no diagnostic data available.</li> </ul>	only with SFC59 (RD_REC) or for SFC13 “DPNRM_DG”
80C1	The data of the previous write job for the same data record on the module have not yet been processed by the module.	–
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	–
80C3	The required resources (memory etc.) are currently occupied.	–
80C4	Communications error: <ul style="list-style-type: none"> <li>Parity error</li> <li>“Software ready” not set</li> <li>Error in field length information</li> <li>Checksum error on the CPU side</li> <li>Checksum error on the module side</li> </ul>	–
80C5	Distributed I/Os not available.	Distributed I/Os
80C6	Data record transfer was stopped due to a priority class abort (restart or background)	Distributed I/Os



## 7.7 Reading a Data Record with SFC59 “RD\_REC” on S7-300 CPUs

### Applicability

The following description of SFC59 “RD\_REC” applies to the CPUs listed below:

CPU	Order Number	Version (and higher)
CPU 312 IFM	6ES7312-5AC00-0AB0	05
CPU 313	6ES7313-1AD00-0AB0	03
CPU 314	6ES7314-1AE01-0AB0	06
CPU 314 IFM	6ES7314-5AE00-0AB0	01
CPU 315	6ES7315-1AF00-0AB0	03
CPU 315-2DP	6ES7315-2AF00-0AB0	03
CPU 614	6ES7614-1AH00-0AB3	06

### Description

With SFC59 “RD\_REC” (read data record), you read the data record with the number RECNUM from the addressed module. The data record read is entered in the destination area indicated by the RECORD parameter providing the data transfer was free of errors.

### Parameters

Table 7-10 Parameters for SFC59 “RD\_REC”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ = 1: Read request
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RECNUM	INPUT	BYTE	I, Q, M, D, L, constant	Data record number (permitted values 0 to 240)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY = 1: Reading is not yet completed.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the data record read. With asynchronous execution of SFC59, make sure that the actual parameters of RECORD have the same length information for all calls. Only data type BYTE is permitted.

**RECORD**

The length information in the output parameter RECORD is interpreted as follows:

Length of the data to be read from the selected data record. This means that the length information of RECORD must not be longer than the actual data record length.

It is advisable to select the length for RECORD exactly the same as the actual data record length.

**Principle of Data Transfer**

With the read job, the CPU informs the addressed module of the length of the RECORD parameter. The following points depend on whether or not the module belongs to a DP station:

- The module is in a central or expansion rack.  
If the length specified by RECORD is shorter than the actual length of the required data record, the CPU reads as many bytes from the start of the data record as specified in the length information of RECORD and enters them in RECORD. RET\_VAL has the value 0.

If the length in RECORD is longer than the actual length of the required data record, the CPU enters an error code in RET\_VAL.

If the length information in RECORD is the same as the actual length of the required data record, the CPU reads the required data record and enters it in RECORD. The value 0 is entered in RET\_VAL.

- The module is located in a DP S7 slave.

The communications processor of the DP S7 slave evaluates the length information received from the CPU. If the length in RECORD is less than the length of the required data record, a DP S7-300 slave returns the required part of the selected data record to the CPU. If the length in RECORD is longer than the length of the required data record, a DP S7-300 slave returns error information to the CPU.

The CPU evaluates the error or length information received from the DP S7 slave.

- If the DP S7 slave provides error information, the corresponding error code is entered in RET\_VAL.
- If the DP S7 slave returns the length of the data read out, this length is compared with the length information in RECORD. Depending on the result of the comparison, an entry is made in the output parameters RET\_VAL and RECORD (The response is the same as when the module is located in a central or expansion rack.)

---

**Note**

With asynchronous processing of SFC59, make sure that the actual parameters of RECORD have the same length information in all calls.

---

## Error Information

Table 7-11 Error Information Specific to SFC59 “RD\_REC”

Error Code (W#16#...)	Explanation	Restriction
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	–
7001	First call with REQ=1: No data transfer active; BUSY has the value 1.	Distributed I/Os
7002	Interim call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/Os
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	–
8093	This SFC is not permitted for the module specified by LADDR and IOID ( the following modules are permitted: S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	–
80A0	Negative acknowledgement when reading from the module (the module was removed during the read job or is defective).	–
80A2	DP protocol error at layer 2	Distributed I/Os
80A3	DP protocol error with user interface/user	Distributed I/Os
80A4	Communication problem on the K bus	The error occurs between the CPU and the external DP interface module.
80B0	<ul style="list-style-type: none"> <li>SFC not possible for module type.</li> <li>The module does not recognize the data record.</li> <li>Data record number <math>\geq 241</math> not permitted.</li> </ul>	–
80B1	The length specified in the RECORD parameter is incorrect.	Length < rec. length
80B2	The configured slot is not occupied.	–
80B3	Actual module type does not match the required module type in SDB1	–
80C0	The module has the data record, but there are still no data to be read.	
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	–
80C3	The required resources (memory etc.) are currently occupied.	–
80C4	Communications error: <ul style="list-style-type: none"> <li>Parity error</li> <li>“Software ready” not set</li> <li>Error in field length information</li> <li>Checksum error on the CPU side</li> <li>Checksum error on the module side</li> </ul>	–
80C5	Distributed I/Os not available.	Distributed I/Os
80C6	Data record transfer was stopped due to a priority class abort (restart or background)	Distributed I/Os

## **7.8 Further Error Information from SFCs 55 to 59**

### **Only for S7-400**

With the S7-400, the SFCs 55 to 59 can also return the error information W#16#80Fx. In this case an error occurred that could not be localized. Please contact the maintenance department in this case.

# SFCs for Handling Time-of-Day Interrupts

## 8

### Chapter Overview

Section	Description	Page
8.1	Handling Time-of-Day Interrupts	8-2
8.2	Characteristics of SFCs 28 to 31	8-3
8.3	Setting a Time-of-Day Interrupt with SFC28 “SET_TINT”	8-5
8.4	Canceling a Time-of-Day Interrupt with SFC29 “CAN_TINT”	8-6
8.5	Activating a Time-of-Day Interrupt with SFC30 “ACT_TINT”	8-7
8.6	Querying a Time-of-Day Interrupt with SFC31 “QRY_TINT”	8-8

## 8.1 Handling Time-of-Day Interrupts

<b>Definition</b>	A time-of-day interrupt results in one of the time-of-day interrupt OBs (OB10 to OB17) being called.
<b>Conditions for the Call</b>	<p>Before a time-of-day interrupt OB can be called by the operating system, the following conditions must be met:</p> <ul style="list-style-type: none"><li>• The time-of-day interrupt OB must have parameters assigned to it (start date and time, execution) using either<ul style="list-style-type: none"><li>– STEP 7,</li><li>or</li><li>– SFC28 “SET_TINT” in the user program.</li></ul></li><li>• The time-of-day interrupt OB must be activated using either<ul style="list-style-type: none"><li>– STEP 7,</li><li>or</li><li>– SFC30 “ACT_TINT” in the user program.</li></ul></li><li>• The time-of-day interrupt OB must not be deselected with STEP 7.</li><li>• The time-of-day interrupt OB must exist in the CPU.</li><li>• If you set the interrupt with SFC30 “ACT_TINT” and if you have specified the execution of the OB as <b>once only</b>, the start date and time must not yet have passed. If you have selected <b>periodic</b> execution, the time-of-day interrupt OB will be called when the next period is completed (start time + multiple of the specified period).</li></ul>
<b>Tip</b>	You can assign parameters to the time-of-day interrupt using STEP 7 and then activate the interrupt in your user program (SFC30 “ACT_TINT”).
<b>Purpose of SFC28 to SFC31</b>	<p>The system functions SFC28 to SFC31 described in the following sections are used as follows:</p> <ul style="list-style-type: none"><li>• To set time-of-day interrupts (SFC28 “SET_TINT”)</li><li>• To cancel time-of-day interrupts (SFC29 “CAN_TINT”)</li><li>• To activate time-of-day interrupts (SFC30 “ACT_TINT”)</li><li>• To query time-of-day interrupts (SFC31 “QRY_TINT”)</li></ul>

## 8.2 Characteristics of SFCs 28 to 31

### What Happens If...

The following table lists a number of different situations and explains the effect they have on a time-of-day interrupt.

If ...	Then ...
A time-of-day interrupt is set (by calling SFC28; SET_TINT)	The current time-of-day interrupt is canceled.
The time-of-day interrupt is canceled (by calling SFC29; CAN_TINT)	The start date and time are cleared. The time-of-day interrupt must then be set again before it can be activated.
The time-of-day interrupt OB does not exist when it is called.	The priority class error is generated automatically, which means that the operating system calls OB85. If OB85 does not exist, the CPU changes to STOP.
The real-time clock is synchronized or the time corrected <ul style="list-style-type: none"> <li>clock adjusted forward</li> <li>clock adjusted back</li> </ul>	If the start date/time is skipped because the clock is moved forward: <ul style="list-style-type: none"> <li>The operating system calls OB80 <sup>1</sup>.</li> <li>Following OB80, every skipped time-of-day interrupt OB is called (once, regardless of the number of periods that were skipped) provided that it was not manipulated in OB80. <sup>2</sup></li> </ul> If OB80 does not exist, the CPU changes to STOP.  If the time-of-day interrupt OBs had already been called during the time by which the clock has been moved back, they are not called again the second time around.

<sup>1</sup> OB80 contains encoded start event information, indicating which time-of-day interrupt OBs could not be called due to moving the clock forward. The time in the start event information corresponds to the time adjusted forward.

<sup>2</sup> The time in the start event information of the time-of-day interrupt activated later after being skipped corresponds to the start time of the first skipped time-of-day interrupt.

**Complete Restart**

During a complete restart, all the time-of-day interrupt settings made in the user program by SFCs are cleared.

The parameters of the “time-of-day interrupts” parameter field set using STEP 7 are then effective.

**Executing the Time-of-Day Interrupt OBs**

The following table shows the different effects of the “execution” parameter. You set this parameter with STEP 7 or with SFC28 “SET\_TINT” (input parameter PERIOD).

Execution of the Time-of-Day Interrupt OBs	Reaction
None (can only be set with STEP 7)	The time-of-day interrupt OB is not executed even when it exists in the CPU. Parameters can be re-assigned in the user program using SFC28 (set time-of-day interrupt).
Once	The time-of-day interrupt is canceled after the time-of-day interrupt OB has been called. It can then be set and activated again.
Periodic (every minute, hour, day, week, month, year)	If the start date and time have already passed when the interrupt is activated, the time-of-day interrupt OB interrupts the cyclic program at the next possible point “start date/time + multiple of the selected period”. In extremely rare situations, processing of the time-of-day interrupt OB may not yet be completed when it is called again. Result: <ul style="list-style-type: none"><li>• Time error, (the operating system calls OB80; if OB80 does not exist, the CPU changes to STOP).</li><li>• The time-of-day interrupt OB is executed later.</li></ul>



### 8.3 Setting a Time-of-Day Interrupt with SFC28 “SET\_TINT”

**Description** With SFC28 “SET\_TINT” (set time-of-day interrupt), you set the start date and time of time-of-day interrupt organization blocks.

#### Parameters

Table 8-1 Parameters for SFC28 “SET\_TINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB started at the time SDT + multiple of PERIOD (OB10 to OB17).
SDT	INPUT	DT	D, L, constant	Start date and time
PERIOD	INPUT	WORD	I, Q, M, D, L, constant	Periods from start point SDT onwards: W#16#0000 = once W#16#0201 = every minute W#16#0401 = hourly W#16#1001 = daily W#16#1202 = weekly W#16#1401 = monthly W#16#1801 = yearly W#16#2001 = at month's end
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.

#### Error Information

Table 8-2 Specific Error Information for SFC28 “SET\_TINT”

Error Code (W#16#...)	Explanation
0000	No error occurred
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past.

## 8.4 Canceling a Time-of-Day Interrupt with SFC29 “CAN\_TINT”

**Description** With SFC29 “CAN\_TINT (cancel time-of-day interrupt), you cancel an activated time-of-day organization block.

### Parameters

Table 8-3 Parameters for SFC29 “CAN\_TINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, in which the start date and time will be canceled (OB10 to OB1).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.

### Error Information

Table 8-4 Specific Error Information for SFC29 “CAN\_TINT”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	No start date/time specified for the time-of-day interrupt OB

## 8.5 Activating a Time-of-Day Interrupt with SFC30 “ACT\_TINT”

**Description** With SFC30 “ACT\_TINT” (activate time-of-day interrupt), you can activate a time-of-day interrupt organization block.

### Parameters

Table 8-5 Parameters for SFC30 “ACT\_TINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OBto be activated (OB10 to OB17).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.

### Error Information

Table 8-6 Specific Error Information for SFC30 “ACT\_TINT”

Error Code (W#16#...)	Explanation
0000	No error occurred
8090	Incorrect parameter OB_NR
80A0	Start date/time not set for the time-of-day interrupt OB
80A1	The activated time is in the past. This error only occurs if execution=once is selected

## 8.6 Querying a Time-of-Day Interrupt with SFC31 “QRY\_TINT”

**Description** Using the system function SFC31 “QRY\_TINT” (query time-of-day interrupt), you can display the status of a time-of-day interrupt organization block at the output parameter STATUS.

### Parameters

Table 8-7 Parameters for SFC31 “QRY\_TINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, whose status will be queried (OB10 to OB17).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the time-of-day interrupt; see Table 8-8.

### Output Parameter STATUS

Table 8-8 Meaning of the Status Bits for SFC31 “QRY\_TINT”

Bit	Value	Meaning
0	0	Time-of-day interrupt is enabled by operating system
1	0	New time-of-day interrupts are accepted
2	0	Time-of-day interrupt is not activated or has elapsed
3	-	-
4	0	Time-of-day interrupt OB is not loaded
5	0	The execution of the time-of-day interrupt OB is disabled by an active test function

### Error Information

Table 8-9 Specific Error Information for SFC31 “QRY\_TINT”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Incorrect parameter OB_NR

# SFCs for Handling Time-Delay Interrupts

# 9

## Chapter Overview

Section	Description	Page
9.1	Handling Time-Delay Interrupts	9-2
9.2	Starting a Time-Delay Interrupt with SFC32 “SRT_DINT”	9-4
9.3	Querying a Time-Delay Interrupt with SFC34 “QRY_DINT”	9-5
9.4	Canceling a Time-Delay Interrupt with SFC33 “CAN_DINT”	9-6

## 9.1 Handling Time-Delay Interrupts

### Definition

After you have called SFC32 “SRT\_DINT”, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the selected time-delay interrupt OB is called. This interrupt is known as a time-delay interrupt.

### Conditions for the Call

Before a time-delay interrupt OB can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by SFC32 “SRT\_DINT”.
- The time-delay interrupt OB must not be deselected with STEP 7.
- The time-delay interrupt OB must exist in the CPU.

### Purpose of SFC32 to SFC34

The system functions SFC32 to SFC34 described in the following sections are used as follows:

- To start time-delay interrupts (SFC32 “SRT\_DINT”)
- To cancel time-delay interrupts (SFC33 “CAN\_DINT”)
- To query time-delay interrupts (SFC34 “QRY\_DINT”)

### What Happens if...

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling SFC32 “SRT_DINT”).	The time-delay interrupt has already started.	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call.	The operating system generates a priority class error (calls OB85). If OB85 does not exist, the CPU changes to STOP.
	The interrupt is started in a start-up OB and the delay time elapses before the CPU changes to RUN.	The call of the time-delay interrupt OB is delayed until the CPU is in the RUN mode.
The delay time has elapsed.	A previously started time-delay interrupt OB is still being executed.	The operating system generates a time error (calls OB80). If OB80 does not exist, the CPU changes to STOP.

**Complete Restart**

During a complete restart, all the time-delay interrupt settings made in the user program by SFCs are cleared.

**Starting in a Start-Up OB**

A time-delay interrupt can be started in a start-up OB (by calling SFC32 “SRT\_DINT” in OB100 or OB101). To call the time-delay interrupt OB, the following two conditions must be met:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in the RUN mode. The time-delay interrupt OB is then called before the first instruction in OB1 is executed.

## 9.2 Starting a Time-Delay Interrupt with SFC32 “SRT\_DINT”

### Description

With SFC32 “SRT\_DINT” (start time-delay interrupt), you start a time-delay interrupt that calls a time-delay interrupt organization block once the delay time has elapsed (parameter DTIME).

With the SIGN parameter, you can enter an identifier that identifies the start of the time-delay interrupt. The values of DTIME and SIGN appear again in the start event information of the specified OB when it is executed.

### Parameters

Table 9-1 Parameters for SFC32 “SRT\_DINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, to be started after a time delay (OB20 to OB23).
DTIME	INPUT	TIME	I, Q, M, D, L, constant	Length of the delay (1 to 60000 ms)
SIGN	INPUT	WORD	I, Q, M, D, L, constant	Identifier that is entered in the start event information of the OB when the time-delay OB is called.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.

### Accuracy

The time between calling SFC32 and the start of the time-delay interrupt OB is a maximum of **one millisecond** less than the selected time providing that no interrupt event delays the call.

### Error Information

Table 9-2 Specific Error Information for SFC32 “SRT\_DINT”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME



### 9.3 Querying a Time-Delay Interrupt with SFC34 “QRY\_DINT”

**Description** With SFC34 “QRY\_DINT” (query time-delay interrupt), you can query the status of a time-delay interrupt OB. Time-delay interrupts are managed by organization blocks OB20 to OB23.

#### Parameters

Table 9-3 Parameters for SFC34 “QRY\_DINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB, whose STATUS will be queried (OB20 to OB23).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being processed, the actual parameter of RET_VAL contains an error code.
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see Table 9-4.

#### Output Parameter STATUS

Table 9-4 Meaning of the Bits of Output Parameter STATUS for SFC34 “QRY\_DINT”

Bit	Value	Meaning
0	0	Time-delay interrupt is enabled by the operating system.
1	0	New time-delay interrupts are not rejected.
2	0	Time-delay interrupt is not activated or has elapsed.
3	–	–
4	0	Time-delay interrupt-OB is not loaded.
5	0	The execution of the time-delay interrupt OB is disabled by an active test function.

#### Error Information

Table 9-5 Specific Error Information for SFC34 “QRY\_DINT”

Error Code (W#16#...)	Explanation
0000	No error occurred
8090	Incorrect parameter OB_NR

## 9.4 Canceling a Time-Delay Interrupt with SFC33 “CAN\_DINT”

### Description

With SFC33 “CAN\_DINT” (cancel time-delay interrupt), you cancel a time-delay interrupt that has already started (see Section 9.2). The time-delay interrupt OB is then not called.

### Parameters

Table 9-6 Parameters for SFC33 “CAN\_DINT”

Parameter	Declaration	Data Type	Memory Area	Description
OB_NR	INPUT	INT	I, Q, M, D, L, constant	Number of the OB to be canceled (OB20 to OB23).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the actual parameter of RET_VAL contains an error code.

### Error Information

Table 9-7 Specific Error Information for SFC33 “CAN\_DINT”

Error Code (W#16#...)	Explanation
0000	No error has occurred.
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.

# SFCs for Handling Synchronous Errors

# 10

## Chapter Overview

Section	Description	Page
10.1	Masking Synchronous Errors	10-2
10.2	Masking Synchronous Errors with SFC36 “MSK_FLT”	10-10
10.3	Unmasking Synchronous Errors with SFC37 “DMSK_FLT”	10-11
10.4	Reading the Error Register with SFC38 “READ_ERR”	10-12

## 10.1 Masking Synchronous Errors

### Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect address areas, numbers or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU “records” the masked errors that have occurred in an error register.

**Unmasking** errors means canceling a previously set mask. Masking is canceled as follows:

- By calling SFC37 “DMSK\_FLT”
- When the currently active OB has been executed.

If an error occurs after it has been unmasked, the operating system starts the corresponding error OB. You can program OB121 for a reaction to programming errors and OB122 for a reaction to access errors.

You can use SFC38 “READ\_ERR” to read out the masked errors that have occurred.

Note: With the S7-300, regardless of whether an error is masked or unmasked, the error is entered in the diagnostic buffer and the group error LED of the CPU is lit.

### Handling Errors in General

If programming and access errors occur in a user program, you can react to them in different ways:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. In this case, the CPU does not change to STOP when an error of this type occurs in the particular priority class. The CPU enters the error in an error register. From this entry, however, you cannot recognize when or how often the error occurred.

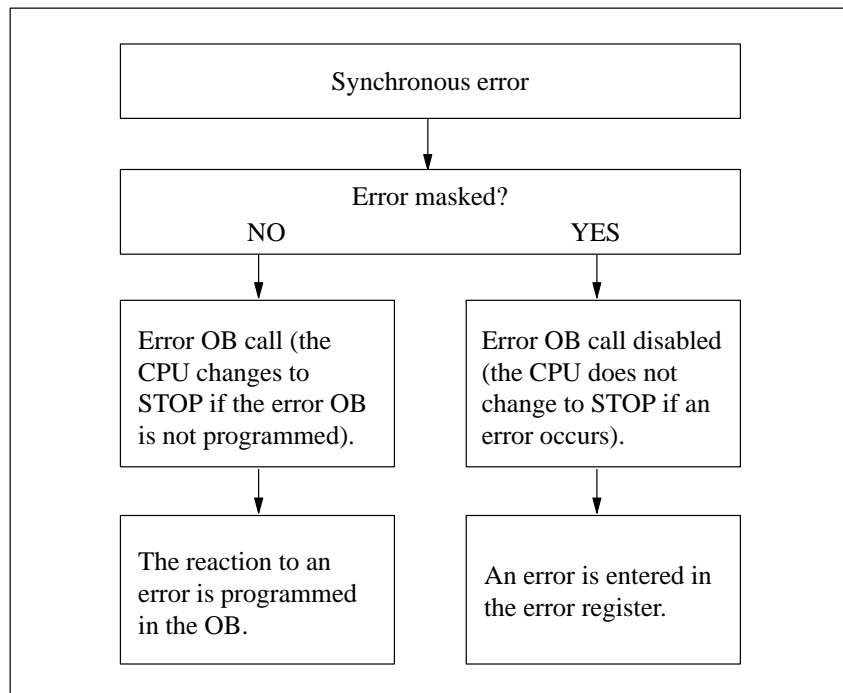


Figure 10-1 Possible Reactions to Synchronous Errors

## Filters

Synchronous errors are assigned to a particular bit pattern known as the **error filter**. This error filter is also in the input and output parameters of SFCs 36, 37 and 38.

The synchronous errors are divided into programming and access errors that you can mask using two error filters. The error filters are illustrated in Figures 10-2 and 10-3.

### Programming Error Filter

Figure 10-2 shows the bit pattern of the error filter for programming errors. The error filter for programming areas is located in the parameters PRGFLT\_... The programming errors are explained in Tables 10-1 and 10-2.

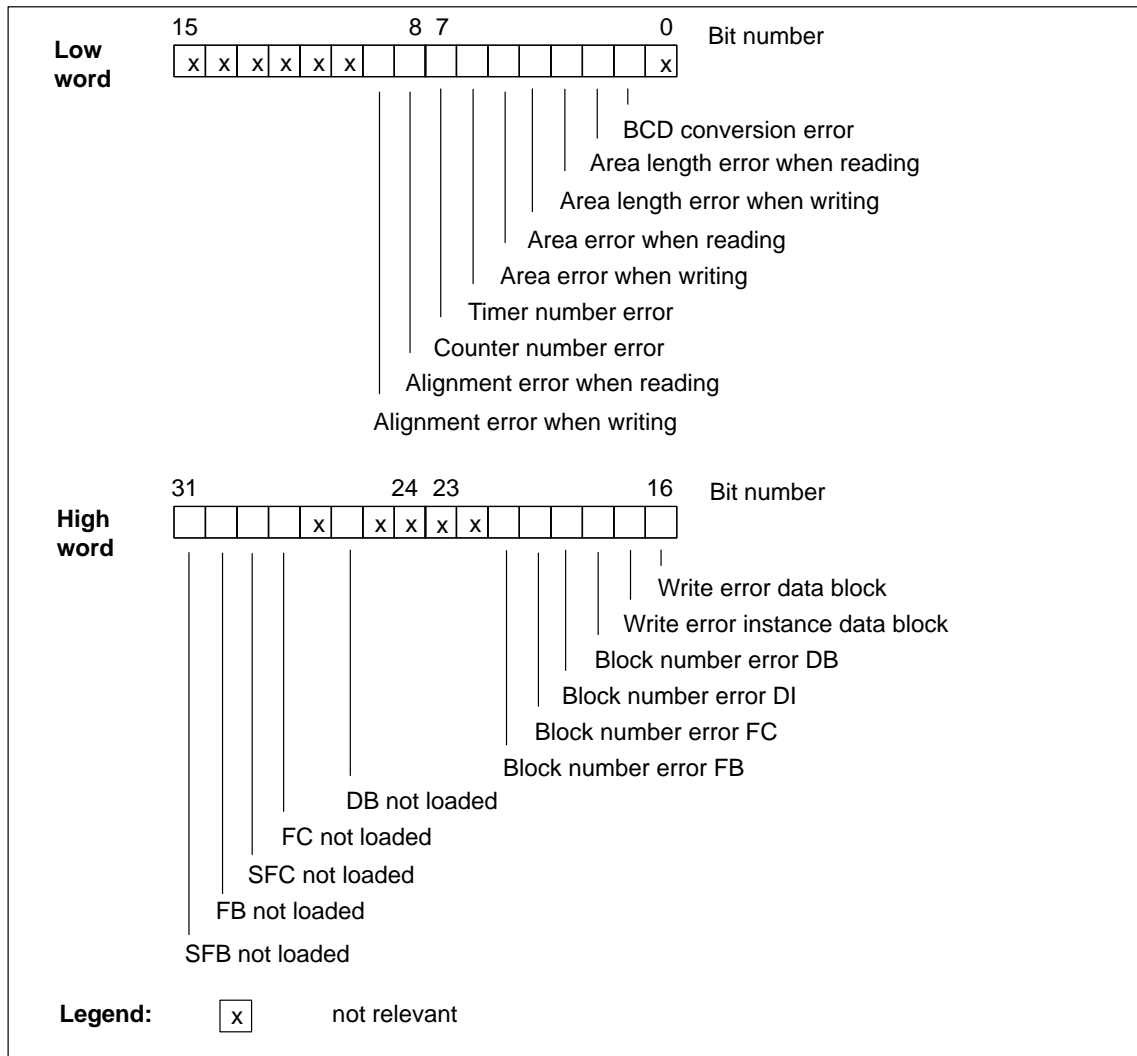


Figure 10-2 Programming Error Filter

### Non-relevant Bits

In Figure 10-2, x stands for ...

- ... Input parameter for SFC36, 37, 38 = "0"
- ... Output parameter for SFC36, 37 = "1" for S7-300  
= "0" for S7-400
- for SFC38 = "0"

Figure 10-3 shows the bit pattern of the error filter for access errors. The error filter for access errors is in the parameters ACCFLT\_... Access errors are explained in Table 10-3.

Figure 10-3 Access Error Filter

In Figure 10-3, **x** means ...

- ... Input parameter for SFC36, 37, 38 = **"0"**
- ... Output parameter for SFC36, 37 = **"1"** with S7-300  
= **"0"** with S7-400
- for SFC38 = **"0"**

**Example**

Figure 10-4 shows the low word of the error filter for access errors with all masked errors:

- As an input parameter for SFC36
- As an output parameter for SFC36

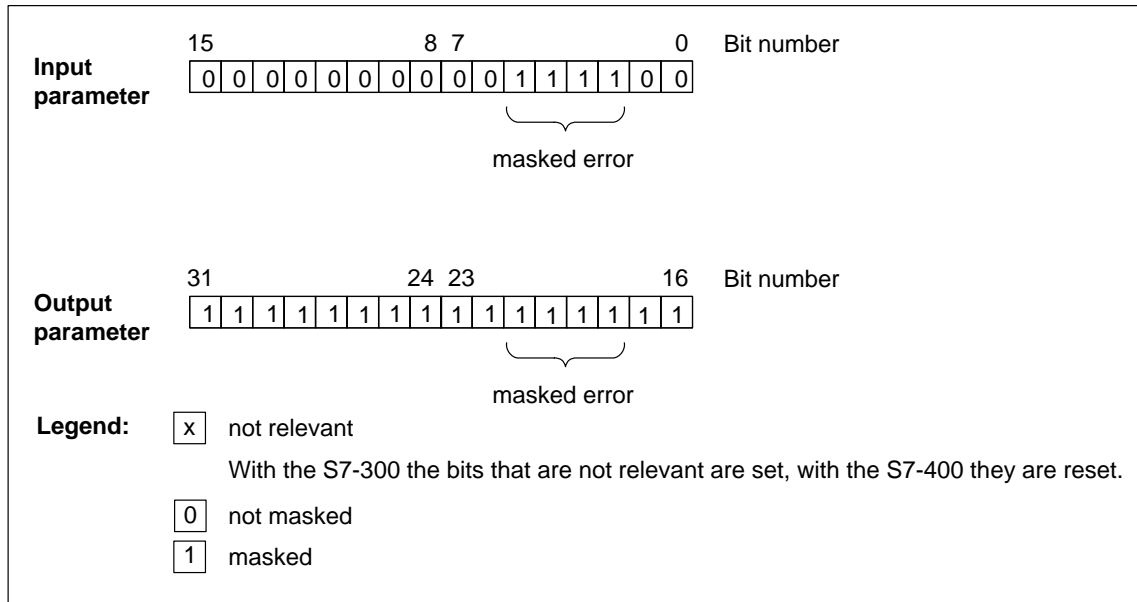


Figure 10-4 Example of an Error Filter



**Programming  
Error Low Word**

Table 10-1 lists the errors assigned to the low word of the error filter for programming errors. The table also shows the possible causes of the errors.

Table 10-1 Possible Causes of Programming Errors, Low Word

Error	Event ID (W#16#...)	Error Occurs ...
BCD conversion error	2521	... when the value to be converted is not a BCD number (for example 5E8)
Area length error when reading	2522	... when an address is being used that is not completely within the possible address area. Example: MW 320 must be read although the memory area is only 256 bytes long.
Area length error when writing	2523	... when an address is being used that is not completely within the possible address area. Example: A value must be written to MW 320 although the memory area is only 256 bytes long.
Area error when reading	2524	... when an incorrect area identifier is specified for the address when using indirect, area-crossing addressing. Example: correct: LAR1 P#E 12.0 L W[AR1, P#0.0] incorrect: LAR1 P#12.0 L W[AR1, P#0.0] For this operation, the area length error is signaled.
Area error when writing	2525	... when an incorrect area identifier is specified for the address when using indirect, area-crossing addressing. Example: correct: LAR1 P#E 12.0 T W[AR1, P#0.0] incorrect: LAR1 P#12.0 T W[AR1, P#0.0] For this operation, the area length error is signaled.
Timer number error	2526	... when a non-existent timer is accessed. Example: SP T [MW 0] where MW 0 = 129; timer 129 must be started although there are only 128 timers available.
Counter number error	2527	... when a non-existent counter is accessed. Example: CU C [MW 0] where MW 0 = 600; counter 600 must be accessed although there are only 512 counters available (CPU 416-D).
Alignment error when reading	2528	... when a byte, word or double word address is addressed with a bit address $\neq 0$ . Example: correct: LAR1 P#M12.0 L B[AR1, P#0.0] incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	... when a byte, word or double word address is addressed with a bit address $\neq 0$ . Example: correct: LAR1 P#M12.0 T B[AR1, P#0.0] incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

# Programming Error High Word

Table 10-2 lists the errors assigned to the high word of the error filter for programming errors. The possible causes of errors are also listed.

Table 10-2 Possible Causes of Programming Errors, High Word

Error	Event ID (W#16#...)	Error Occurs ...
Write error data block	2530	... when the data block to be written to is read only.
Write error instance data block	2531	... when the instance data block to be written to is read only.
Block number error DB	2532	... when a data block must be opened whose number is higher than the highest permitted number.
Block number error DI	2533	... when an instance data block must be opened whose number is higher than the highest permitted number.
Block number error FC	2534	... when a function is called whose number is higher than the highest permitted number.
Block number error FB	2535	... when a function block is called whose number is higher than the highest permitted number.
DB not loaded	253A	... when the data block to be opened is not loaded.
FC not loaded	253C	... when the called function is not loaded.
SFC does not exist	253D	... when the called system function does not exist.
FB not loaded	253E	... when the function block to be called is not loaded.
SFB not existing	253F	... when the called system/standard function block is not existing.

**Access Errors**

Tabelle 10-3 lists the errors assigned to the error filter for access errors. The possible causes of the errors are also listed.

Table 10-3 Possible Causes of Access Errors

Error	Event ID (W#16#...)	for ...	Error Occurs ...
I/O access error when reading	2942	S7-300	... when no signal module is assigned to the address in the I/O area. or ... when access to this I/O area is not acknowledged within the selected module watchdog time (time-out).
		S7-400	... during the first incorrect read access (time-out)
	2944	S7-400	... when no signal module is assigned to the address in the I/O area, or during the nth incorrect read access ( $n > 1$ ).
I/O access error when writing	2943	S7-300	... when no signal module is assigned to the address in the I/O area. or ... when access to this I/O area is not acknowledged within the selected module watchdog time (time-out).
		S7-400	... during the first incorrect write access (time-out)
	2945	S7-400	... when no signal module is assigned to the address in the I/O area or during the nth incorrect write access ( $n > 1$ ).

## 10.2 Masking Synchronous Errors with SFC36 “MSK\_FLT”

### Description

With SFC36 “MSK\_FLT” (mask synchronous errors), you can control the reaction of the CPU to synchronous errors. With this SFC, you can mask the synchronous errors using the error filter (see Section 10.1). When you call SFC36, you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to “1” in the input parameters, other bits that were set previously retain their value “1”. You therefore obtain new error filters that you can read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You can read out the error register with SFC38 “READ\_ERR” (see Section 10.4).

### Parameters

Table 10-4 Parameters of SFC36 “MSK\_FLT”

Parameter	Declaration	Data Type	Memory Area	Description
PRGFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Programming error to be masked
ACCFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Access error to be masked
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked program errors
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Masked access errors

### Error Information

Table 10-5 Specific Error Information for SFC36 “MSK\_FLT”

Error Code (W#16#...)	Explanation
0000	None of the errors was already masked.
0001	At least one of the errors was already masked. Nevertheless the other errors will be masked.

### 10.3 Unmasking Synchronous Errors with SFC37 “DMSK\_FLT”

#### Description

With SFC37 “DMSK\_FLT” (unmask synchronous errors), you unmask the errors masked with SFC36 “MSK\_FLT”. To do this, you must set the corresponding bits of the error filter to “1” in the input parameters (see Section 10.1). With the SFC37 call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the entries are cleared in the error register. You can read out the new error filters using the output parameters.

#### Parameters

Table 10-6 Parameters of SFC37 “DMSK\_FLT”

Parameter	Declaration	Data Type	Memory Area	Description
PRGFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Programming errors to be unmasked
ACCFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, constant	Access errors to be unmasked
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Still masked access errors

#### Error Information

Table 10-7 Specific Error Information for SFC37 “DMSK\_FLT”

Error Code (W#16#...)	Explanation
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless the other errors will be unmasked.

## 10.4 Reading the Error Register with SFC38 “READ\_ERR”

### Description

Using SFC38 “READ\_ERR” (read error register), you can read the error register. The structure of the error register corresponds to that of the programming and access error filters which you can program as input parameters with SFC36 and SFC37 (see Section 10.1).

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call SFC38, you read the required entries from the error register and at the same time clear the entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

### Parameters

Table 10-8 Parameters for SFC38 “READ\_ERR”

Parameter	Declaration	Data Type	Memory Area	Description
PRGFLT_QUERY	INPUT	DWORD	I, Q, M, D, L, constant	Query programming errors
ACCFLT_QUERY	INPUT	DWORD	I, Q, M, D, L, constant	Query access errors
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
PRGFLT_ESR	OUTPUT	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_ESR	OUTPUT	DWORD	I, Q, M, D, L	Access errors that occurred

### Error Information

Table 10-9 Specific Error Information for SFC38 “READ\_ERR”

Error Code (W#16#...)	Explanation
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.

# SFCs for Handling Interrupts and Asynchronous Errors

# 11

## Chapter Overview

Section	Description	Page
11.1	Delaying and Disabling Interrupts and Asynchronous Errors	11-2
11.2	Disabling the Processing of New Interrupts and Asynchronous Errors with SFC39 “DIS_IRT”	11-4
11.3	Enabling the Processing of New Interrupts and Asynchronous Errors with SFC40 “EN_IRT”	11-6
11.4	Delaying the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC41 “DIS_AIRT”	11-8
11.5	Enabling the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC42 “EN_AIRT”	11-9

## 11.1 Delaying and Disabling Interrupts and Asynchronous Errors

### Purpose of SFC39 to SFC42

With these SFCs, you can achieve the following:

- Disable interrupts and asynchronous errors with SFC39 “DIS\_IRT” for all subsequent CPU cycles.
- Delay higher priority classes with SFC41 “DIS\_AIRT” until the end of the OB.
- Enable interrupts and asynchronous errors with SFC40 “EN\_IRT” or SFC42 “EN\_AIRT”.

You program the handling of interrupts and asynchronous errors in the user program. You must also program the corresponding OBs.

### Advantage of SFC41 and SFC42

Delaying higher priority interrupts and asynchronous errors by disabling them with SFC41 “DIS\_AIRT” and then enabling them again with SFC42 “EN\_AIRT” has the following advantages:

The number of interrupts delayed is counted by the CPU. If you have delayed interrupts and asynchronous errors, the delay cannot be canceled by standard FC calls if the interrupts and asynchronous errors are also disabled and then enabled again in the standard FCs themselves.

### Interrupt Classes

The interrupts are divided into various classes. Table 11-1 lists all the interrupt classes and the corresponding OBs.

Table 11-1 Interrupt Classes in STEP 7 and the Corresponding OBs

Interrupt Class	OB
Time-of-day interrupts	OB10 to OB17
Time-delay interrupts	OB20 to OB23
Cyclic interrupts	OB30 to OB38
Hardware interrupts	OB40 to OB47
Communication interrupts	OB50, OB51
Multicomputing interrupt	OB60
Asynchronous error interrupts	OB80 to OB87 (see Table 11-2)
Synchronous error interrupts	OB121, OB122 (You can mask or unmask the processing of synchronous error interrupts with SFC36 to SFC38)



**Asynchronous Errors**

Table 11-2 lists all the asynchronous errors to which you can react with an OB call in the user program.

Table 11-2 Asynchronous Errors and the Corresponding OBs

<b>Asynchronous Errors</b>	<b>OB</b>
Time error (for example, cycle time exceeded)	OB80
Power supply error (for example, battery fault)	OB81
Diagnostic interrupt (for example, defective fuse on a signal module)	OB82
Remove/insert module interrupt	OB83
CPU hardware fault (for example, memory card removed)	OB84
Program error	OB85
Rack failure	OB86
Communication error	OB87

## 11.2 Disabling the Processing of New Interrupts and Asynchronous Errors with SFC39 “DIS\_IRT”

### Description

With SFC39 “DIS\_IRT” (disable interrupt), you disable the processing of new interrupts and asynchronous errors. This means that if an interrupt occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB or asynchronous error OB,
- **nor** triggers the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous errors, this remains in effect for all priority classes. The effects of “DIS\_IRT” can only be canceled again by calling SFC40 “EN\_IRT” (see Section 11.3) or by a complete restart.

Regardless of whether or not they are disabled, the operating system writes interrupts and asynchronous errors to the diagnostic buffer when they occur.

### Note

Remember that when you program the use of SFC39 “DIS\_IRT”, all interrupts that occur are lost!

### Parameters

Table 11-3 Parameters for SFC39 “DIS\_IRT”

Parameter	Declaration	Data Type	Memory Area	Description
MODE	INPUT	BYTE	I, Q, M, D, L, constant	Specifies which interrupts and asynchronous errors are disabled (see Table 11-4).
OB_NR	INPUT	INT	I, Q, M, D, L, constant	OB number
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

### MODE

Table 11-4 Permitted Values for the MODE Parameter of SFC39 “DIS\_IRT”

MODE	Meaning
0	Newly occurring interrupts and asynchronous errors are disabled. (Synchronous errors are not disabled.)
1	All newly occurring events belonging to a specified interrupt class are disabled. You specify the interrupt class by specifying the number of the first OB, for example OB40 for hardware interrupts (see Table 11-1).
2	All new occurrences of a specified interrupt are disabled. You specify the interrupt using the OB number.

**Error Information**

Table 11-5 Specific Error Information for SFC39 “DIS\_IRT”

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8090	The input parameter OB_NR contains an illegal value.
8091	The input parameter MODE contains an illegal value.

### 11.3 Enabling the Processing of New Interrupts and Asynchronous Errors with SFC40 “EN\_IRT”

#### Description

With SFC40 “EN\_IRT” (enable interrupt), you enable the processing of new interrupts and asynchronous errors that you previously disabled with SFC39 “DIS\_IRT”. This means that if an interrupt event occurs, the operating system of the CPU reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
- It triggers the standard reaction if the interrupt OB or asynchronous error OB is not programmed.

#### Parameters

Table 11-6 Parameters for SFC40 “EN\_IRT”

Parameter	Declaration	Data Type	Memory Area	Description
MODE	INPUT	BYTE	I, Q, M, D, L, constant	Specifies which interrupts and asynchronous errors will be enabled (see Table 11-8).
OB_NR	INPUT	INT	I, Q, M, D, L, constant	OB number
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active the return value contains an error code.

#### MODE

Table 11-7 Permitted Values for the MODE Parameter of SFC40 “EN\_IRT”

MODE	Meaning
0	All newly occurring interrupts and asynchronous errors are enabled.
1	All newly occurring events of a specified interrupt class are enabled. You specify the interrupt class using the number of the first OB, for example OB40 for hardware interrupts (see Table 11-1).
2	All newly occurring events of a specified interrupt are enabled. You specify the interrupt using the OB number.

**Error Information**

Table 11-8 Specific Error Information for SFC40 “EN\_IRT”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	The input parameter OB_NR contains an illegal value.
8091	The input parameter MODE contains an illegal value.

## 11.4 Delaying the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC41 “DIS\_AIRT”

### Description

With SFC41 “DIS\_AIRT” (disable alarm interrupts), you delay the processing of interrupt OBs and asynchronous error OBs which have a higher priority than that of the current OB. You can call SFC41 more than once in an OB. The SFC41 calls are counted by the operating system. Each of these calls remains in effect until it is canceled again specifically by an SFC42 “EN\_AIRT” call **or** until the current OB has been completely processed.

Once they are enabled again, the interrupts and asynchronous errors that occurred while SFC41 was in effect are processed as soon as they are enabled again with SFC42 “EN\_AIRT” or as soon as the current OB has been executed.

### Parameters

Table 11-9 Parameter for SFC41 “DIS\_AIRT”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of delays (= number of SFC41 calls)

### Return Value

Table 11-10 shows the return value for SFC41 that is output with the RET\_VAL parameter.

Table 11-10 Return Value of SFC41 “DIS\_AIRT”

Return Value	Description
n	”n” shows the number of times that processing was disabled, in other words the number of SFC41 calls (interrupt processing is only enabled again when n = 0; see Section 11.5).

## 11.5 Enabling the Processing of Higher Priority Interrupts and Asynchronous Errors with SFC42 “EN\_AIRT”

**Description** With SFC42 “EN\_AIRT” (enable alarm interrupts), you enable the processing of higher priority interrupts and asynchronous errors that you previously disabled with SFC41 “DIS\_AIRT”. Each SFC41 call must be canceled by an SFC42 call.

**Example** If, for example, you have disabled interrupts five times with five SFC41 calls, you must cancel these calls with five SFC42 calls.

### Parameters

Table 11-11 Parameter for SFC42 “EN\_AIRT”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of delays still programmed on completion of SFC42 or error message.

**Return Value and Error Information** How you evaluate the error information of the RET\_VAL parameter is explained in Chapter 2. This chapter also contains the general error information for the SFCs. Table 11-12 contains all the error information specific to SFC42 that can be output with the RET\_VAL parameter.

Table 11-12 Return Value and Error Information for SFC42 “EN\_AIRT”

Return Value and Error Information	Description
n	“n” shows the number of SFC41 calls not yet canceled by SFC42 calls (interrupt processing is only enabled again when “n” = 0).
W#16#8080	The function has been called again although interrupt processing was already enabled.





# SFCs for Diagnostics

# 12

## Chapter Overview

Section	Description	Page
12.1	Reading OB Start Information with SFC6 “RD_SINFO”	12-2
12.2	Reading a System Status List or Partial List with SFC51 “RDSYSST”	12-4
12.3	Writing a User-Defined Diagnostic Event to the Diagnostic Buffer with SFC52 “WR_USMSG”	12-11

## System Diagnostics

The CPUs maintain internal data about the status of the programmable logic controller. With the system diagnostics functions, you can read out the most important data. Some of the data can be displayed on the programming device using STEP 7.

You can also access the data required for system diagnostics in your program, by using the SFCs “RD\_SINFO” and “RDSYSST”.

## 12.1 Reading OB Start Information with SFC6 “RD\_SINFO”

### Description

With SFC6 “RD\_SINFO” (read start information), you can read the start information about the following:

- The last OB to be called that has not yet been completely executed
- The last start-up OB to be started.

There is no time stamp in either case. If the call is in OB100 or OB101, two identical start information messages are returned.

### Parameters

Table 12-1 Parameters for SFC6 “RD\_SINFO”

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
TOP_SI	OUTPUT	STRUCT	D, L	Start information of the current OB
START_UP_SI	OUTPUT	STRUCT	D, L	Start information of the start-up OB last started

### TOP\_SI and START\_UP\_SI

The output parameters TOP\_SI and START\_UP\_SI are two structures with identical elements (see Table 12-2).

Table 12-2 Elements of the Structures TOP\_SI and START\_UP\_SI

Structure Element	Data Type	Description
EV_CLASS	BYTE	<ul style="list-style-type: none"> <li>• Bits 0 to 3: Event ID</li> <li>• Bits 4 to 7: Event class</li> </ul>
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Number of the priority class
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: identifies the information entered in ZI2_3
TYP1	BYTE	Data ID 1: identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Bits 4 to 7 of the EV\_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB (see Chapter 2).

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the start-up OB that was started last.

**Error Information**

SFC6 "RD\_SINFO" does not provide any specific error information but only general error information. The general error codes and how to evaluate them are described in detail in Chapter 2.

## 12.2 Reading a System Status List or Partial List with SFC51 “RDSYSST”

### Description

With system function SFC51 “RDSYSST” (read system status), you read a system status list or a partial system status list.

You start the reading by assigning the value 1 to the input parameter REQ when SFC51 is called. If the system status could be read immediately, the SFC returns the value 0 at the BUSY output parameter. If BUSY has the value 1, the read function is not yet completed. (see Section 2.2).

### Note

If you call SFC51 “RDSYSST” in the diagnostic interrupt OB with the SZL-ID W#16#00B1 or W#16#00B2 or W#16#00B3 and access the module that initiated the diagnostic interrupt, the system status is read immediately.

### System Resources

If you start several asynchronous read functions (the jobs with SZL\_ID W#16#00B4 and W#16#4C91 and W#16#4092 and W#16#4292 and W#16#4692 and possibly W#16#00B1 and W#16#00B3) one after the other at brief intervals, the operating system ensures that all the read jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET\_VAL. You can remedy this temporary error situation by repeating the job.

The maximum number of “simultaneously” active SFC51 jobs depends on the CPU. You will find this information in /70/ and /101/.

### Parameters

Table 12-3 Parameters for SFC51 “RDSYSST”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ=1: Starts processing
SZL_ID	INPUT	WORD	I, Q, M, D, L, constant	SZL-ID of the system status list or partial list to be read (the partial lists are explained in Appendix B).
INDEX	INPUT	WORD	I, Q, M, D, L, constant	Type or number of an object in a partial list.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while executing the SFC, the RET_VAL parameter contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	TRUE: Reading not yet completed.
SZL_HEADER	OUTPUT	STRUCT	D, L	See below.
DR	OUTPUT	ANY	I, Q, M, L, D	Field of data records read.

**SZL\_HEADER**

The SZL\_HEADER parameter is a structure defined as follows:

```
SZL_HEADER: STRUCT
    LENGTHDR:  WORD
    N_DR:      WORD
END_STRUCT
```

LENGTHDR is the length in bytes of a data record in the field of data records read.

N\_DR is the number of data records in the field of data records read.

**Error Information**

Table 12-4 Specific Error Information for SFC51 “RDSYSST”

Error Code (W#16#...)	Description
0000	No error
0081	Result field too short. (Nevertheless as many data records as possible are supplied. The SLZ header indicates this number.)
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.
7002	Interim call (REQ irrelevant): Data transfer already active; BUSY has the value 1.
8081	Result field too short (not enough space for one data record).
8082	SZL_ID is wrong or is unknown in the CPU or SFC.
8083	INDEX wrong or not permitted.
8085	Due to a problem in the system, information is not currently available (for example due to a lack of resources).
8086	The data record cannot be read due to a system error (bus, modules, operating system).
8087	Data record cannot be read because the module does not exist or does not acknowledge.
8088	Data record cannot be read because the actual type identifier is different from the expected type identifier.
8089	Data record cannot be read because the module is not capable of diagnostics.
808A	Data type is not permitted for the DR parameter (the data types BOOL, BYTE, CHAR, WORD, DWORD, INT, DINT are permitted) or the bit address is not 0.
80A2	DP protocol error (layer 2 error) (temporary error)
80A3	DP protocol error with user interface/user (temporary error)
80A4	Communication problem on K bus (error occurs between the CPU and the external DP interface module)
80C5	Distributed I/Os not available (temporary error).
80C6	Data record transfer stopped due to priority class abort (restart or background)

**SZL\_IDs****Note**

Refer to /70/ and /101/ for the SZL\_IDs available in your specific CPU. For the partial lists that can be read out with SFC51 “RDSYSST” refer to

- /70/ for the S7-300
- The following table for the S7-400.

Table 12-5 Partial System Status Lists that Can Be Read Out with SFC51 (S7-400)

<b>SZL_ID (W#16#...)</b>	<b>Partial List</b>	<b>INDEX (W#16#...)</b>
	<b>Module ID</b>	
0011	All identification data records	Irrelevant
0111	One identification data record	0001
0F11	Only SZL partial list header information	Irrelevant
	<b>CPU characteristics</b>	
0012	All characteristics	Irrelevant
0112	Characteristics of one group	
	MC7 processing unit	0000
	Time system	0100
	System behavior	0200
	MC7 language description	0300
0F12	Only SZL partial list header information	Irrelevant
	<b>User memory areas</b>	
0013	Data records of the work memory and of the memory for CFBs assigned in the system	Irrelevant
0113	One data record for the memory area specified	
	Work memory	0001
	Size of memory for CFBs assigned in the system	0006
0F13	Only SZL partial list header information	Irrelevant
	<b>System areas</b>	
0014	Data records of all system areas	Irrelevant
0114	Data records of one system area	
	Process-image input (number of bytes)	0001
	Process-image output (number of bytes)	0002
	Number of memory markers	0003
	Number of timers	0004
	Number of counters	0005
	User data P area (number of bytes for logical address area)	0006
	Total local data area of the CPU in bytes	0007
0F14	Only SZL partial list header information	Irrelevant

Table 12-5 Partial System Status Lists that Can Be Read Out with SFC51 (S7-400), continued

SZL_ID (W#16#...)	Partial List	INDEX (W#16#...)
	<b>Module types</b>	
0015	Data records of all module types	Irrelevant
0115	Data record of one module type	
	OBs	0800
	DBs	0A00
	SDBs	0B00
	FCs	0C00
	FBs	0E00
0F15	Only SZL partial list header information	Irrelevant
	<b>Existing priority classes</b>	
0016	Data records of all priority classes	Irrelevant
0116	One data record for the priority class specified	
	Free scan cycle	0000
	Time-of-day interrupt	000A
	Time-delay interrupt	0014
	Cyclic interrupt	001E
	Hardware interrupt	0028
	Multiprocessor interrupt	003C
	Asynchronous error interrupt	0050
	Background cycle	005A
	Startup	0064
	Synchronous error interrupt	0078
0F16	Only SZL partial list header information	Irrelevant
	<b>Interrupt / error assignment</b>	
0021	Data records of all possible interrupts	Irrelevant
0121	Data records of all interrupts possible in one interrupt class	Same as for SZL_ID W#16#0116
0221	Data record of the interrupt specified	OB No.
0921	Data records of all interrupts of one interrupt class for which the interrupt OB is loaded	Same as for SZL_ID W#16#0116
0A21	Data records of all interrupts for which the interrupt OB is loaded	Irrelevant
0F21	Only SZL partial list header information	Irrelevant
	<b>Interrupt status</b>	
0122	Data records of all interrupts possible in one interrupt class	Same as for SZL_ID W#16#0116
0222	Data record of the interrupt specified.	OB number
0822	Data records of all interrupts of one interrupt class for which the interrupt OB is loaded	Same as for SZL_ID W#16#0116
0F22	Only SZL partial list header information	Irrelevant
	<b>Status of the priority classes</b>	
0123	Data record of one priority class	Priority class
0223	Data records of the priority classes being executed	Irrelevant

Table 12-5 Partial System Status Lists that Can Be Read Out with SFC51 (S7-400), continued

SZL_ID (W#16#...)	Partial List	INDEX (W#16#...)
0F23	Only SZL partial list header information	Irrelevant
	<b>Operating modes</b>	
0124	Information about last operating mode change	Irrelevant
0424	Currently active operating mode	Irrelevant
	<b>Communication status data</b>	
0132	Status data for one communication unit	
	Diagnostics	0005
	Time system	0008
	<b>Start information list</b>	
0281	Start information of all synchronous error OBs of one priority class	Priority class
0381	Start information of all OBs of one priority class	Priority class
0681	Start information of all synchronous error OBs of one priority class before execution	Priority class
0781	Start information of all OBs of one priority class before execution	Priority class
0A81	Start information of all synchronous error OBs of one priority class being executed	Priority class
0B81	Start information of all OBs of one priority class being executed	Priority class
0F81	Only SZL partial list header information	Irrelevant
	<b>Start event list</b>	
0082	All start events	Irrelevant
0282	Start events of all currently active synchronous error OBs of one priority class	Priority class
0382	Start events of all OBs of one priority class	Priority class
0682	Start events of all synchronous error OBs of one priority class before execution	Priority class
0782	Start events of all OBs of one priority class before execution	Priority class
0A82	Start events of all synchronous error OBs of one priority class being executed	Priority class
0B82	Start events of all OBs of one priority class being executed	Priority class
0F82	Only SZL partial list header information	Irrelevant
	<b>Module status information</b> (a maximum of 27 data records is supplied)	
0091	Status information of all modules / submodules inserted	Irrelevant
0191	Module status information of all modules / racks with incorrect type ID	Irrelevant
0291	Module status information of all faulty modules	Irrelevant
0391	Module status information of all unobtainable modules	Irrelevant
0991	Module status information of all submodules of the host module in the rack specified	Rack or DP master system ID



Table 12-5 Partial System Status Lists that Can Be Read Out with SFC51 (S7-400), continued

SZL_ID (W#16#...)	Partial List	INDEX (W#16#...)
0C91	Module status information of a module in a central configuration or connected to an integrated DP communications processor	Logical base address
4C91	Module status information of a module connected to an external DP communications processor	Logical base address
0D91	Module status information of all modules in the rack / DP station specified	Rack or DP master system ID or DP master system ID and station number
0E91	Module status information of all modules allocated	Irrelevant
0F91	Only SZL partial list header information	Irrelevant
	<b>Rack/station status information</b>	
0092	Expected status of the rack in the central configuration / of the stations of a DP master system	0 / DP master system ID
4092	Expected status of the stations of a DP master system connected to an external DP interface	DP master ID
0292	Actual status of the rack in the central configuration / of the stations of a DP master system	0 / DP master system ID
4292	Actual status of the stations of a DP master system connected via an external DP interface module	DP master system ID
0392	Status of the battery backup of the racks in a central configuration	irrelevant
0492	Status of the entire backup of the racks in a central configuration	irrelevant
0592	Status of the 24V power supply for the racks in a central configuration	irrelevant
0692	OK state of the expansion racks in a central configuration / of the stations of a DP master system connected via an integrated DP interface module	0 / DP Master system ID
4692	OK state of the stations of a DP master system connected via an external DP interface module	DP master system ID
0F92	Only SZL partial list header information	irrelevant
	<b>Diagnostic buffer</b> (a maximum of 21 data records is supplied)	
00A0	All entries that can be supplied in the currently active operating mode	Irrelevant
01A0	The most recent entries, the number is specified in the index	Quantity
04A0	Start information of all standard OBs	Irrelevant
05A0	All entries of communication units	Irrelevant
06A0	All entries of the object management system	Irrelevant
07A0	All entries from test and installation	Irrelevant

Table 12-5 Partial System Status Lists that Can Be Read Out with SFC51 (S7-400), continued

<b>SZL_ID (W#16#...)</b>	<b>Partial List</b>	<b>INDEX (W#16#...)</b>
08A0	All operating mode execution information	Irrelevant
09A0	All entries caused by asynchronous errors	Irrelevant
0AA0	All entries caused by synchronous errors	Irrelevant
0BA0	All entries caused by STOP, abort, operating mode transitions	Irrelevant
0CA0	All entries caused by H/F events	Irrelevant
0DA0	All diagnostic entries	Irrelevant
0EA0	All user entries	Irrelevant
0FA0	Only SZL partial list header information	Irrelevant
	<b>Diagnostic data on modules</b>	
00B1	The first four diagnostic bytes of one module (data record 0)	Logical base address
00B2	All diagnostic data of one module ( $\leq 220$ bytes, data record 1) (no DP module)	Rack, slot
00B3	All diagnostic data of one module ( $\leq 220$ bytes, data record 1)	Logical base address
00B4	Diagnostic data of a DP slave	Configured diagnostic address

## 12.3 Writing a User-Defined Diagnostic Event to the Diagnostic Buffer with SFC52 “WR\_USMSG”

### Description

With SFC52 “WR\_USMSG” (write user element in diagnostic buffer), you write a user-defined diagnostic event to the diagnostic buffer. You can also send the corresponding diagnostic message to all stations logged on for this purpose. If an error occurs, the output parameter RET\_VAL provides the error information.

### Sending a User-Defined Diagnostic Message

SFC52 writes a user-defined diagnostic event to the diagnostic buffer. You can then also send the corresponding diagnostic message to any station logged on for this purpose. The user-defined diagnostic message is then written to the send buffer and automatically sent to the logged on stations.

You can check whether the sending of user-defined diagnostic messages is currently possible. To do this, call SFC51 “RDSYSST” with the parameters SZL\_ID = W#16#0132 and INDEX = W#16#0005. The fourth word of the data record obtained indicates whether sending a user element is currently possible (1) or not (0).

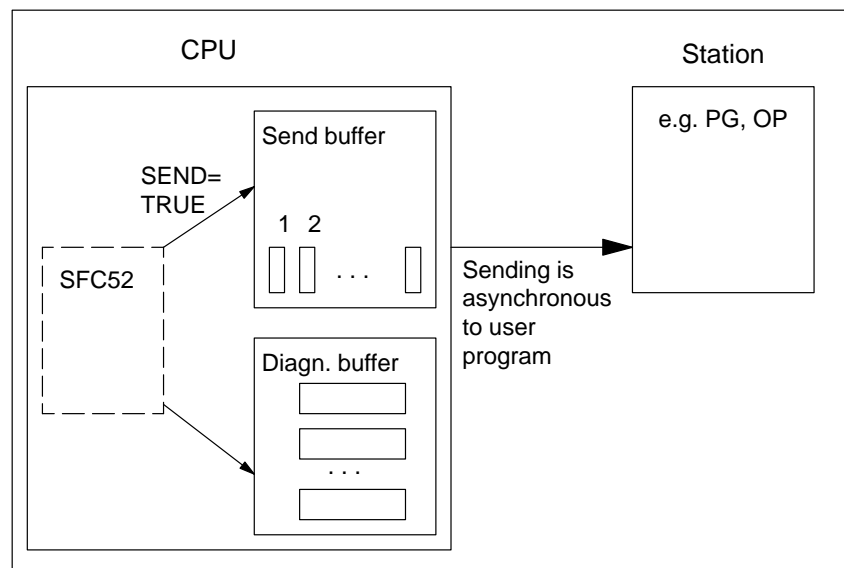


Figure 12-1 Sending a User-Defined Diagnostic Message to a Logged on Station

**Send Buffer Full**

The diagnostic message can only be entered in the send buffer if the send buffer is not full. The number of entries that can be made in the send buffer depends on the type of CPU you are using.

If the send buffer is full, then:

- The diagnostic event is nevertheless entered in the diagnostic buffer,
- The parameter RET\_VAL indicates that the send buffer is full (RET\_VAL = W#16#8092).

**Station Not Logged On**

If a user-defined diagnostic message is to be sent (SEND = TRUE) and no station is logged on,

- The user-defined diagnostic event is entered in the diagnostic buffer,
- the parameter RET\_VAL indicates that no station is logged on (RET\_VAL = W#16#8091).

**General Structure**

The internal structure of an element in the diagnostic buffer is as follows:

Table 12-6 Structure of an Entry in the Diagnostic Buffer

Byte	Content
1 and 2	Event ID
3	Priority class
4	OB number
5 and 6	Reserved
7 and 8	Additional information 1
9, 10, 11 and 12	Additional information 2
13 to 20	Time stamp

**Event ID**

The structure of the event ID is explained in Section C.1. An event ID is assigned to every event.

**Additional Information**

This is additional information about the event. The additional information can be different for each event. When you create a diagnostic event, you can decide on the content of these entries yourself.

**Time Stamp**

The time stamp is of the type Date\_and\_Time.

## Parameters

Table 12-7 Parameters of SFC52 “WR\_USMSG”

Parameter	Declaration	Data Type	Memory Area	Description
SEND	INPUT	BOOL	I, Q, M, D, L, constant	Enable the sending of the user-defined diagnostic message to all logged-on stations
EVENTN	INPUT	WORD	I, Q, M, D, L, constant	Event ID - You assign the event ID. This is not assigned by the message server.
INFO1	INPUT	ANY	I, Q, M, D, L	Additional information 1 word long
INFO2	INPUT	ANY	I, Q, M, D, L	Additional information 2 words long
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information

### SEND

If SEND = TRUE, the user-defined diagnostic message is sent to all logged-on stations. The message is only sent if the station is logged on and if the send buffer is not full. The sending of the element is asynchronous to the user program.

### EVENTN

The EVENTN parameter contains the event ID of the user-defined diagnostic event. You can enter event IDs of the types W#16#8xyz, W#16#9xyz, W#16#Axyz, W#16#Bxyz. The structure of the event ID is explained in Section C.1.

### INFO1

The INFO1 parameter contains information that is one word long. The following data types are permitted for INFO1:

- WORD
- INT
- ARRAY [0 to 1] OF CHAR

### INFO2

The INFO2 parameter contains information that is two words long. The following data types are permitted for INFO2:

- DWORD
- DINT
- REAL
- TIME
- ARRAY [0 to 3] OF CHAR

**Error Information**

Table 12-8 Specific Error Information for SFC52 “WR\_USRMSG”

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error
8083	Data type of INFO1 not permitted
8084	Data type of INFO2 not permitted
8085	EVENTN not permitted
8086	Length of INFO1 not permitted
8087	Length of INFO2 not permitted
8091	No station logged on (entry made in diagnostic buffer)
8092	Sending not possible at present, send buffer full (diagnostic event entered in the diagnostic buffer)

# SFCs for Updating the Process Image and Processing Bit Fields

# 13

## Chapter Overview

Section	Description	Page
13.1	Updating the Process Image Input Table with SFC26 “UPDAT_PI”	13-2
13.2	Updating the Process Image Output Table with SFC27 “UPDAT_PO”	13-3
13.3	Setting a Range of Outputs with SFC79 “SET”	13-4
13.4	Resetting a Range of Outputs with SFC80 “RSET”	13-5
13.5	Implementing a Sequencer with SFB32 “DRUM”	13-6

## 13.1 Updating the Process Image Input Table with SFC26 “UPDAT\_PI”

### Description

With SFC26 “UPDAT\_PI” (update process image), you update the entire process image input table or a section of it. You must first define the section of the process image with STEP 7.

### Note

Each logical address you assign to a section of the process image input table with STEP 7 no longer belongs to the entire process image input table.

The updating of the entire process image input table by the system at the start of the cyclic program is influenced by SFC26 calls.

### Parameters

Table 13-1 Parameters for SFC26 “UPDAT\_PI”

Parameter	Declaration	Data Type	Memory Area	Description
PART	INPUT	BYTE	I, Q, M, D, L, constant	Number of the process image section to be updated. Permitted values: 0 to 8 (0 means entire process image, n where $1 \leq n \leq 8$ means process image section n)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
FLADDR	OUTPUT	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error occurred.

### Error Information

Table 13-2 Specific Error Information for SFC26 “UPDAT\_PI”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for the input parameter PART.
8091	The specified process image section was not defined.
80A0	An access error was detected during the updating.



## 13.2 Updating the Process Image Output Table with SFC27 “UPDAT\_PO”

### Description

With SFC27 “UPDAT\_PO” (update process outputs), you transfer the signal states of the entire process image output table or a section of it to the output modules. The process image section must first be defined with STEP 7.

### Note

Each logical address you assign to a section of the process image output table with STEP 7 no longer belongs to the entire process image output table.

The transfer of the entire process image output table to the output modules at the end of the cyclic program is not influenced by SFC27 calls.

### Parameters

Table 13-3 Parameters for SFC27 “UPDAT\_PO”

Parameter	Declaration	Data Type	Memory Area	Description
PART	INPUT	BYTE	I, Q, M, D, L, constant	Number of the process image section to be transferred. Permitted values: 0 to 8 (0 means entire process image, n where $1 \leq n \leq 8$ means process image section n)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
FLADDR	OUTPUT	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error occurred.

### Error Information

Table 13-4 Specific Error Information for SFC27 “UPDAT\_PO”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for the input parameter PART.
8091	The specified process image section was not defined.
80A0	An access error was detected during updating.

### 13.3 Setting a Range of Outputs with SFC79 “SET”

#### Description

When the master control relay is set, calling SFC79 “SET” (set range of outputs) has the following effect:

- The bit field in the peripheral I/O area selected with the parameters N and SA is set.
- The corresponding bits in the process image output table are also set.

The bit field must be the part of the peripheral I/O area assigned to a process image.

If no module is plugged in for part of the selected bit field, SFC79 still attempts to set the entire bit field. It then returns the appropriate error information in RET\_VAL.

If the master control relay is not set, calling SFC79 has no effect.

When SFC79 is executed, whole bytes are written in the peripheral I/O area. If the bit field selected with the parameters N and SA does not begin or end at a byte boundary, calling SFC79 has the following effect:

- The bits in the first and last bytes to be transferred to the peripheral I/O area and that do not belong to the selected bit field contain the value of the corresponding bits in the process image output table.
- The bits belonging to the selected bit field are set as explained above.

If you assign the value 0 to the N parameter, calling SFC79 has no effect.

#### Parameter

Table 13-5 Parameters for SFC79 “SET”

Parameter	Declaration	Data Type	Memory Area	Description
N	INPUT	INT	I, Q, M, D, L, constant	Number of bits to be set
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
SA	OUTPUT	POINTER	P	Pointer to the first bit to be set

#### Error Information

How you evaluate the error information of the parameter RET\_VAL is explained in Chapter 2. This chapter also contains the general error information of the SFCs. SFC79 does not provide any specific error information with the RET\_VAL parameter.

## 13.4 Resetting a Range of Outputs with SFC80 “RSET”

### Description

When the master control relay is set, calling SFC80 “RSET” (reset range of outputs) has the following effect:

- The bit field in the peripheral I/O area selected with the parameters N and SA is reset.
- The corresponding bits in the process image output table are also reset.

The bit field must be located in the part of the peripheral I/O area to which a process image is assigned.

If no module is plugged in for part of the selected bit field, SFC80 still attempts to reset the entire bit field. It then returns the appropriate error information in RET\_VAL.

If the master control relay is not set, calling SFC80 has no effect.

When SFC80 is executed, whole bytes are written to the peripheral I/O area. If the bit field selected with the parameters N and SA does not begin or end at a byte boundary, calling SFC80 has the following effect:

- The bits in the first and last bytes to be transferred to the peripheral I/O area and that do not belong to the selected bit field contain the value of the corresponding bits in the process image output table.
- The bits belonging to the selected bit field are set as explained above.

If you assign the value 0 to the N parameter, calling SFC80 has no effect.

### Parameters

Table 13-6 Parameters for SFC80 “RSET”

Parameter	Declaration	Data Type	Memory Area	Description
N	INPUT	INT	I, Q, M, D, L, constant	Number of bits to be reset
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
SA	OUTPUT	POINTER	P	Pointer to the first bit to be reset

### Error Information

How you evaluate the error information of the parameter RET\_VAL is explained in Chapter 2. This chapter also contains the general error information of the SFCs. SFC80 does not provide any specific error information with the RET\_VAL parameter.

## 13.5 Implementing a Sequencer With SFB32 "DRUM"

### Description

SFB32 "DRUM" implements a sequencer with a maximum of 16 steps. You specify the number of the first step with the DSP parameter and the number of the last step with the LST\_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT\_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT\_VAL array that you specify or the output bit is assigned the value of the corresponding output bit of the previous step. Which value is assigned depends on how you set the mask bits in the S\_MASK parameter (see Table 13-7).

SFB32 "DRUM" switches to the next step when there is a rising edge at the JOG input compared with the previous SFB call. If the SFB has already reached the last step, a rising edge at JOG sets the variables Q and EOD; DCC has the value 0; and the SFB remains in the last step until 1 is set at the RESET input.

You can also assign parameters so that switching to the next step is time dependent. To do this, you must set the DRUM\_EN parameter to 1. The sequencer then switches to the next step when:

- The event bit EVENTi is set for the current step and
- The time programmed for the current step has expired.

This time is the product of the DTBP time base and the time factor valid for the current step (from the S\_PRESET array)

---

### Note

The execution time remaining in the current step (DCC) is only reduced when the corresponding event bit EVENTi is set.

---

If a 1 is set at the RESET input when the SFB is called, the sequencer goes to the step you assigned to the DSP input.

---

### Note

If you set a 1 for DRUM\_EN, you can achieve the following special situation:

- Purely time-dependent enabling of the steps by selecting  $\text{EVENT}_i = 1$  where  $\text{DSP} \leq i \leq \text{LST\_STEP}$ .
  - Purely event-dependent enabling of the steps by setting 0 at DTBP.
- 

When the sequencer is in the last step (DSC has the value LST\_STEP) and when the execution time for this step has expired, outputs Q and EOD are set and the SFB remains in the last step until you set 1 at the RESET input.

A DRUM timer runs only in the STARTUP and RUN modes.

The operating system does not reset SFB32 “DRUM” during a complete restart. If you want to initialize SFB32 “DRUM” after a complete restart, call it with RESET = 1 in OB100.

## Parameters

Table 13-7 Parameters for SFB 32 “DRUM”

Parameter	Declaration	Data Type	Memory Area	Description
RESET	INPUT	BOOL	I, Q, M, D, L, constant	Signal level 1 resets the sequencer.
JOG	INPUT	BOOL	I, Q, M, D, L, constant	A rising edge (compared to the last SFB call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assign to DRUM_EN.
DRUM_EN	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter that specifies whether time-dependent switching to the next step is possible (1: time-dependent switching possible)
LST_STEP	INPUT	BYTE	I, Q, M, D, L, constant	Number of the last step possible values: 1 to 16
EVENT <sub>i</sub> , 1 ≤ i ≤ 16	INPUT	BOOL	I, Q, M, D, L, constant	Event bit no. i (belongs to step i)
OUT <sub>j</sub> , 0 ≤ j ≤ 15	OUTPUT	BOOL	I, Q, M, D, L	Output bit no. j (identical to the bit no. j of OUT_WORD)
Q	OUTPUT	BOOL	I, Q, M, D, L	Status parameter that specifies whether the execution time you selected for the last step has expired.
OUT_WORD	OUTPUT	WORD	I, Q, M, D, L, P	Output bits collected together in a variable
ERR_CODE	OUTPUT	WORD	I, Q, M, D, L, P	If an error occurs during execution of the SFB, ERR_CODE contains the error information.
JOG_HIS	VAR	BOOL	I, Q, M, D, L, constant	(No relevance for the user: JOG input parameter of the previous SFB call)
EOD	VAR	BOOL	I, Q, M, D, L, constant	Identical to the output parameter Q
DSP	VAR	BYTE	I, Q, M, D, L, P, constant	Number of the first step possible values: 1 to 16
DSC	VAR	BYTE	I, Q, M, D, L, P, constant	Number of the current step
DCC	VAR	DWORD	I, Q, M, D, L, P, constant	The execution time still remaining in the current step in ms (only relevant if DRUM_EN = 1 and the corresponding event bit is set to = 1)
DTBP	VAR	WORD	I, Q, M, D, L, P, constant	The time base valid for all steps in ms

Table 13-7 Parameters for SFB 32 “DRUM”, continued

Parameter	Declaration	Data Type	Memory Area	Description
PREV_TIME	VAR	DWORD	I, Q, M, D, L, constant	(Not relevant for the user: system time of the previous SFB call)
S_PRESET	VAR	ARRAY of WORD	I, Q, M, D, L, constant	One-dimensional array with the time factor for each step. A sensible selection of the indices would be: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.
OUT_VAL	VAR	ARRAY of BOOL	I, Q, M, D, L, constant	Two-dimensional array with the values output in each step if they have not been masked out using S_MASK. A sensible selection for the indices would be: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	VAR	ARRAY of BOOL	I, Q, M, D, L, constant	Two-dimensional array with the mask bits for each step. A sensible selection of the indices would be: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the y(th) value to be output in step x. Meaning of the mask bits: 0: The value of the previous step is assigned to the corresponding output bit. 1: The corresponding value from OUT_VAL is assigned to the corresponding output bit.

**Error Information**

If one of the conditions listed in the following table occurs, SFB32 “DRUM” remains in its current status and the ERR\_CODE output is set.

Table 13-8 Possible Values of the ERR\_CODE Output Parameter

ERR_CODE (W#16#...)	Explanation
0000	No error
8081	Illegal value for LST_STEP
8082	Illegal value for DSC
8083	Illegal value for DSP
8084	The product $DCC = DTBP * S\_PRESET[DSC]$ exceeds the value $2^{**32}-1$ (approximately 24.86 days)

# System Functions for Addressing Modules **14**

## Chapter Overview

Section	Description	Page
14.1	Querying the Logical Address of a Channel with SFC5 “GADR_LGC”	14-2
14.2	Querying the Module Slot belonging to a Logical Address with SFC49 “LGC_GADR”	14-4
14.3	Querying all Logical Addresses of a Module with SFC50 “RD_LGADR”	14-6

## 14.1 Querying the Logical Address of a Channel with SFC5 “GADR\_LGC”

### Description

Based on the channel of a signal module, the corresponding module slot and the offset user data address area of the module are known. With SFC5 “GADR\_LGC” (convert geographical address to logical address), you can obtain the corresponding logical address.

### Parameters

Table 14-1 Parameters for SFC5 “GADR\_LGC”

Parameter	Declaration	Data Type	Memory Area	Description
SUBNETID	INPUT	BYTE	I, Q, M, D, L, constant	Area identifier: <ul style="list-style-type: none"> <li>0, if the slot is in one of the racks 0 (central rack) or 1 to 21 (expansion rack)</li> <li>DP master ID of the corresponding distributed I/O system if the slot is in a distributed I/O device.</li> </ul>
RACK	INPUT	WORD	I, Q, M, D, L, constant	<ul style="list-style-type: none"> <li>No. of the rack if the area identifier is 0</li> <li>Station number of the distributed I/O device if the area identifier &gt; 0</li> </ul>
SLOT	INPUT	WORD	I, Q, M, D, L, constant	Slot number
SUBSLOT	INPUT	BYTE	I, Q, M, D, L, constant	Submodule slot (if no submodule can be plugged in, 0 must be specified here)
SUBADDR	INPUT	WORD	I, Q, M, D, L, constant	Offset in the user data address area of the module
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
IOID	OUTPUT	BYTE	I, Q, M, D, L	Area identifier: B#16#54: peripheral input (PI) B#16#55: peripheral output (PQ) If the module is a mixed module, the SFC supplies the area identifier B#16#54.
LADDR	OUTPUT	WORD	I, Q, M, D, L	Logical address of the channel



**Error Information**

Table 14-2 Specific Error Information for SFC5 "GADR\_LGC"

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID.
8095	Illegal value for the RACK parameter.
8096	Illegal value for the SLOT parameter.
8097	Illegal value for the SUBSLOT parameter.
8098	Illegal value for the SUBADDR parameter.
8099	The slot is not configured.
809A	The subaddress of the selected slot is not configured.

## 14.2 Querying the Module Slot belonging to a Logical Address with SFC49 “LGC\_GADR”

**Description** With SFC49 “LGC\_GADR” (convert logical address to geographical address), you obtain the module slot belonging to a logical address and the offset in the user data address area of the module.

### Parameters

Table 14-3 Parameters for SFC49 “LGC\_GADR”

Parameter	Declaration	Data Type	Memory Area	Description
IOID	INPUT	BYTE	I, Q, M, D, L, constant	ID of the address area: B#16#54 = Peripheral input (PI) B#16#55 = Peripheral output (PQ) If the module is a mixed module, specify the area ID of the lowest address. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the module. With mixed modules, specify the lower of the two addresses.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
AREA	OUTPUT	BYTE	I, Q, M, D, L	Area ID: this specifies how the remaining output parameters must be interpreted.
RACK	OUTPUT	WORD	I, Q, M, D, L	Rack number
SLOT	OUTPUT	WORD	I, Q, M, D, L	Slot number
SUBADDR	OUTPUT	WORD	I, Q, M, D, L	Offset in the user data address area of the corresponding module.

**Output Parameter AREA**      The output parameter AREA specifies how the output parameters RACK, SLOT and SUBADDR must be interpreted (see Table 14-4).

Table 14-4      Meaning of RACK, SLOT and SUBADDR Dependent on the Value of AREA

Value of AREA	System	Meaning of RACK, SLOT and SUBADDR		
0	S7-400	RACK	△	Rack number
		SLOT	△	Slot number
1	S7-300	SUBADDR	△	Difference between the logical address and logical base address
2	DP	RACK, SLOT and SUBADDR have no significance.		
3	S5 P area	RACK	△	Rack number
4	S5 O area		△	
5	S5 IM3 area		△	
6	S5 IM4 area		△	
		SLOT	△	Slot number of the adapter casing
		SUBADDR	△	Address in the S5 x area

## Error Information

Table 14-5      Specific Error Information for SFC49 "LGC\_GADR"

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter

### 14.3 Querying all Logical Addresses of a Module with SFC50 “RD\_LGADR”

#### Description

You start with one logical address of a module. With SFC50 “RD\_LGADR” (read module logical addresses), you obtain all the declared logical addresses of this module. You have already assigned addresses to modules previously with STEP 7. SFC50 enters the logical addresses obtained in the field PEADDR or in the field PAADDR in ascending order.

#### Parameters

Table 14-6 Parameters for SFC50 “RD\_LGADR”

Parameter	Declaration	Data Type	Memory Area	Description
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Area identifier: B#16#54: peripheral input (PI) B#16#55: peripheral output (PQ)
LADDR	INPUT	WORD	I, Q, M, D, L, constant	One logical address
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
PEADDR	OUTPUT	ANY	I, Q, M, D, L	Field for the PI addresses, field elements must be of the WORD data type.
PECOUNT	OUTPUT	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	OUTPUT	ANY	I, Q, M, D, L	Field for the PQ addresses, field must be of the WORD data type.
PACOUNT	OUTPUT	INT	I, Q, M, D, L	Number of returned PQ addresses

**Error Information**

Table 14-7 Specific Error Information for SFC50 “RD\_LGADR”

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter
80A0	Error in the output parameter PEADDR: The data type of the field elements is not WORD.
80A1	Error in the output parameter PAADDR: The data type of the field elements is not WORD.
80A2	Error in the output parameter PEADDR: The specified field could not accommodate all the logical addresses.
80A3	Error in the output parameter PAADDR: The specified field could not accommodate all the logical addresses.



## SFCs for Distributed I/Os

# 15

### Chapter Overview

Section	Description	Page
15.1	Triggering a Hardware Interrupt on the DP Master with SFC7 “DP_PRAL”	15-2
15.2	Reading Diagnostic Data of a DP Slave with SFC13 “DPNRM_DG” (Slave Diagnostics)	15-4
15.3	Reading Consistent Data of a DP Standard Slave with SFC14 “DPRD_DAT”	15-7
15.4	Writing Consistent Data to a DP Standard Slave with SFC15 “DPWR_DAT”	15-9

## 15.1 Triggering a Hardware Interrupt on the DP Master with SFC7 “DP\_PRAL”

### Applicability

You can only use the SFC described in this section, if you are using a CPU 315-2DP as an intelligent slave.D

### Description

With SFC7 “DP\_PRAL”, you trigger a hardware interrupt on the DP master from the user program of an intelligent slave. This interrupt starts OB40 on the DP master.

Using the input parameter AL\_INFO, you can identify the cause of the hardware interrupt. This interrupt identifier is transferred to the DP master and you can evaluate the identifier in OB40 (variable OB40\_POINT\_ADDR) (see Section 1.6).

The requested hardware interrupt is specified uniquely by the input parameters IOID and LADDR. For each configured address area in the transfer memory, you can trigger exactly one hardware interrupt at any time.

### How the SFC Functions

SFC7 “DP\_PRAL” operates asynchronously, in other words, it is executed over several SFC calls. You start the hardware interrupt request by calling SFC7 with REQ=1. The status of the job is indicated by the output parameters RET\_VAL and BUSY, see Section 2.2. The job is completed when execution of OB40 is completed on the DP master.

---

#### Note

If you operate the DP slave as a standard slave, the job is completed as soon as the diagnostic frame is fetched by the DP master.

---

### Identifying a Job

The input parameters IOID and LADDR specify the job unequivocally. If you have called SFC7 “DP\_PRAL” on a DP slave and you call this SFC again before the master has acknowledged the requested hardware interrupt, the way in which the SFC reacts depends largely on whether the new call involves the same job: if the parameters IOID and LADDR match a job that is not yet completed, the SFC call is interpreted as a follow-on call regardless of the value of the parameter AL\_INFO and the value W#16#7002 is entered in RET\_VAL.



## Parameters

Table 15-1 Parameters for SFC 7 “DP\_PRAL”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ=1: Trigger hardware interrupt on the DP master belonging to the slave
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier of the address range in the transfer memory (from the point of view of the DP slave): B#16#54= Peripheral input (PI) B#16#55= Peripheral output (PQ) The identifier of a range belonging to a mixed module is the lower of the two addresses. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Start address of the address range in the transfer memory (from the point of view of the DP slave). If this is a range belonging to a mixed module, specify the lower of the two addresses.
AL_INFO	INPUT	DWORD	I, Q, M, D, L, constant	Interrupt ID This is transferred to the OB40 that will be started on the DP master (variable OB40_POINT_ADDR). If you operate the intelligent slave with a remote master, you must evaluate the diagnostic frame on the master. (see /70/ )
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains an error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: The triggered hardware interrupt has not yet been acknowledged by the DP master.

## Error Information

Table 15-2 Error Information Specific to SFC7 “DP\_PRAL”

Error Code (W#16#...)	Explanation
0000	The job was executed without errors.
7000	First call with REQ=0. No hardware interrupt request is active; BUSY has the value 0.
7001	First call with REQ=1. A hardware interrupt request has already been sent to the DP master; BUSY has the value 1.
7002	Interim call (REQ irrelevant): the triggered hardware interrupt has not yet been acknowledged by the DP master; BUSY has the value 1.
8090	Start address of the address range in the transfer memory
8091	Interrupt is blocked (block configured by user)
8093	The parameters IOID and LADDR address a module that is not capable of a hardware interrupt.
80C6	Distributed peripheral I/Os not currently available.

## 15.2 Reading Diagnostic Data of a DP Slave with SFC13 “DPNRM\_DG” (Slave Diagnostics)

### Slave Diagnostics

Each DP slave provides slave diagnostic data structured in accordance with EN 50 170 Volume 2, PROFIBUS. To read out this diagnostic data, you require SFC13 “DPNRM\_DG”.

Refer to the following table for the basic structure of the slave diagnostic data and to the manuals of the DP slaves for further information.

Table 15-3 Structure of Slave Diagnostic Data

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Further slave-specific diagnostic information

### Description

With SFC13 “DPNRM\_DG” (read diagnostic data of a DP slave), you read the current diagnostic data of a DP slave in the format specified by EN 50 170 Volume 2, PROFIBUS. The data that has been read is entered in the destination area indicated by RECORD following error-free data transfer. You start the read job by assigning 1 to the input parameter REQ in the SFC13 call.

**Function** The read job is executed asynchronously, in other words it requires several SFC13 calls. The status of the job is indicated by the output parameters RET\_VAL and BUSY, see Section 2.2.

## Parameters

Table 15-4 Parameters for SFC13 “DPNRM\_DG”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	REQ=1: Read request
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured diagnostic address of the DP slave.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code. If no error occurred, the length of the data actually transferred is entered in RET_VAL.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the diagnostic data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the destination area is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostic data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the destination area and the overflow bit is set in the data.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: The read job is not yet completed.

## Input Parameter RECORD

The CPU evaluates the actual length of the diagnostic data that were read as following:

- If the length information of RECORD is less than the number of data bytes supplied, the data are discarded and a corresponding error code is entered in RET\_VAL.
- If the length information of RECORD is longer or equal to the number of supplied data bytes, the data are accepted in the destination area and the actual length is entered in RET\_VAL as a positive value.

## Note

You must make sure that the actual parameters of RECORD match in all calls belonging to a job.

A job is identified unequivocally by the LADDR input parameter.

**Standard Slaves  
with more than  
240 Bytes of  
Diagnostic Data**

With standard slaves on which the number of standard diagnostic data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data are discarded and the corresponding error information is entered in RET\_VAL.
- is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostic data are transferred to the destination area and the overflow bit is set in the data.

**Output Parameter  
RET\_VAL**

- If an error occurs while the function is being executed, the return value contains an error code.
- If no error occurs during the data transfer, RET\_VAL contains the length of the data read in bytes as a positive number.

**Error Information**

How you evaluate the error information of the RET\_VAL parameter is explained in Chapter 2. This chapter also contains the general error information for the SFCs. The error information specific to SFC13 is a subset of the error information for SFC59 "RD\_REC", see Table 7-9.

**System Resources  
for S7-400**

When SFC13 "DPNRM\_DG" is called for a job that is not currently being processed, resources of the CPU (memory space) are occupied on the S7-400. You can call SFC13 in quick succession for several DP slaves providing that you do not exceed the maximum number of "simultaneously" active SFC13 jobs for your CPU. You will find the maximum number of such jobs in **/101/**. If you activate several jobs "simultaneously", all the jobs will be executed without interfering with each other. If you reach the limits of the system resources, this is indicated in RET\_VAL. In this case, repeat the job.

### 15.3 Reading Consistent Data of a DP Standard Slave with SFC14 “DPRD\_DAT”

**Definition:**  
**Consistent Data** Data that belong together and that must not be separated are known as consistent data. The values, for example, from analog modules must always be handled as consistent data, in other words, the value of an analog module must not be falsified by reading out at two different times.

**Purpose of SFC14** You require SFC14 “DPRD\_DAT” because you can only read out a maximum of four continuous bytes using load instructions that access the I/Os or the process image input table.

**Description** With SFC14 “DPRD\_DAT” (read consistent data of a DP standard slave), you read the consistent data of a DP standard slave. The length must be three or more than four bytes, with the maximum length being fixed for each specific CPU. You will find the maximum length in the technical data of your CPU. If no error occurred during the data transfer, the data that have been read are entered in the destination area identified by RECORD. The destination area must have the same length as configured for the selected module with STEP 7. If you read from a DP standard slave with a modular design or with several DP identifiers, you can only access the data of one module/DP identifier per SFC14 call specifying the configured start address.

#### Parameters

Table 15-5 Parameters for SFC14 “DPRD\_DAT”

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the I area of the module from which the data will be read.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Destination area for the user data that were read. This must be exactly as long as you configured for the selected module with STEP 7. Only the BYTE data type is permitted.

**Error Information**

Table 15-6 Specific Error Information for SFC14 “DPRD\_DAT”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8090	You have not configured a module for the specified logical base address or you have ignored the restriction concerning the length of consistent data.
8092	A type other than BYTE is specified in the ANY reference.
9093	No DP module from which you can read consistent data exists at the logical address specified in LADDR.
80A0	The selected module is incorrect.
80B1	The length of the specified destination area is not identical to the user data length configured with STEP 7.

## 15.4 Writing Consistent Data to a DP Standard Slave with SFC15 “DPWR\_DAT”

**Definition:** Data that belong together and that must not be separated are known as consistent data. The values, for example, from analog modules must always be handled as consistent data, in other words, the value of an analog module must not be falsified by reading out at two different times.

**Purpose of SFC15** You require SFC15 “DPWR\_DAT” because you can only write a maximum of four continuous bytes using the transfer instructions that access the I/Os or the process image input table.

**Description** With SFC15 “DPWR\_DAT” (write consistent data to a DP standard slave), you transfer the data in RECORD consistently to the addressed DP standard slave. The length must be three or more than four bytes, with the maximum length being fixed for each specific CPU. You will find this information in the technical data for your CPU. The data is transferred synchronously, in other words, on completion of the SFC, the write job is also completed. The source area must have the same length as you configured for the selected module with STEP 7. If the DP standard slave has a modular design, you can only access one module of the DP slave.

### Parameters

Table 15-7 Parameters for SFC15 “DPWR\_DAT”

Parameter	Declaration	Data Type	Memory Area	Description
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Configured start address from the output area of the module to which the data will be written.
RECORD	INPUT	ANY	I, Q, M, D, L	Source area for the user data to be written. This must be exactly as long as you configured for the selected module with STEP 7. Only the BYTE data type is permitted.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is active, the return value contains an error code.

**Error Information**

Table 15-8 Specific Error Information for SFC15 “DPWR\_DAT”

<b>Error Code (W#16#...)</b>	<b>Explanation</b>
0000	No error occurred.
8090	You have not configured a module for the specified logical base address or you have ignored the restriction concerning the length of consistent data.
8092	A type other than BYTE is specified in the ANY reference.
9093	No DP module to which you can write consistent data exists at the logical address specified in LADDR.
80A1	The selected module is incorrect.
80B1	The length of the specified source area is not identical to the user data length configured with STEP 7.



## SFCs for Global Data Communication

# 16

### Chapter Overview

Section	Description	Page
16.1	Sending a GD Packet with SFC60 “GD_SND”	16-2
16.2	Fetching a Received GD Packet with SFC61 “GD_RCV”	16-4

## 16.1 Sending a GD Packet with SFC60 “GD\_SND”

### Description

With SFC60 “GD\_SND” (global data send), the data of a GD packet are collected and then sent on the path specified in the GD packet. The GD packet must already have been configured with STEP 7.

SFC60 “GD\_SND” can be called at any point in the user program.

The scan rate and the collection and sending of the data by the system at the cycle checkpoint are not influenced by SFC60 calls.

### Interruptability

SFC60 “GD\_SND” can be interrupted by higher priority classes. It is also possible that SFC60 is called again for the same GD packet in the higher priority class.

The data are then collected and sent in the higher priority class. When the program returns to the interrupted SFC, this is terminated immediately and the data that have already been collected are discarded.

This procedure means that during the processing of the highest priority class, consistent data are transferred (consistency in the sense defined for global data).

### Data Consistency with GD

The following rules apply to the consistency of the data collected from the various memory areas and sent.

The following are consistent:

- The simple data types (bit, byte, word and double word)
- An array of the data types byte, word and double word up to a maximum length depending on the specific CPU.

### Ensuring Consistency for an Entire GD Packet

A GD packet on the CPU sending the data has a structure that does not automatically guarantee that the collected data are consistent. This is, for example, the case when the packet consists of an array of bytes and the number of bytes exceeds the maximum length for the specific CPU.

If, however, you require consistency for the entire GD packet, follow the procedure below in your program:

- Disable or delay the occurrence of higher priority interrupts and asynchronous errors by calling SFC39 “DIS\_IRT” or SFC41 “DIS\_AIRT”.
- Call SFC60 “GD\_SND”.
- Enable the higher priority interrupts and asynchronous errors again by calling SFC40 “EN\_IRT” or SFC42 “EN\_AIRT”.

## Parameters

Table 16-1 Parameters for SFC60 “GD\_SND”

Parameter	Declaration	Data Type	Memory Area	Description
CIRCLE_ID	INPUT	BYTE	I, Q, M, D, L, constant	Number of the GD circle in which the GD packet to be sent is located. You specify this number when configuring the global data with STEP 7. Permitted values: 1 to 16. The maximum number of possible GD circles can be found in the technical data of your CPU.
BLOCK_ID	INPUT	BYTE	I, Q, M, D, L, constant	Number of the GD packet to be sent in the selected GD circle. This number is set during configuration of the global data by STEP 7. Permitted values: 1 to 3. The maximum number of possible GD circles can be found in the technical data of your CPU.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information

## Error Information

Table 16-2 Specific Error Information for SFC60 “GD\_SND”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8081	The GD packet selected with the parameters CIRCLE_ID and BLOCK_ID is not configured.
8082	Illegal value for the parameters CIRCLE_ID or BLOCK_ID or for both parameters.
8083	An error occurred during the execution of the SFC. The type of error is entered in the variable configured for the status information. This can be evaluated by your program.
8084	The execution of the SFC was terminated prematurely because SFC60 was called again for the same GD packet in a higher priority class (see “Interruptability”).
8085	An error occurred entering the status information in the configured variable.

### Note

Following each SFC60 call, you should evaluate the corresponding GD packet status and, if necessary, reset it.

## 16.2 Fetching a Received GD Packet with SFC61 “GD\_RCV”

<b>Description</b>	<p>With SFC61 “GD_RCV” (global data receive), the data from an incoming GD frame for exactly one GD packet are fetched and entered in the receive GD packet. This must already have been configured with STEP 7.</p> <p>SFC61 “GD_SND” can be called at any point in the user program.</p> <p>The scan rate and the fetching of the data by the system at the cycle checkpoint are not influenced by SFC61 calls.</p>
<b>Interruptability</b>	<p>SFC61 “GD_SND” can be interrupted by higher priority classes, however, only so that the data consistency defined for global data remains guaranteed. If the processing of the function is interrupted, it is possible that SFC61 is called again for the same GD packet in the higher priority class.</p> <p>The data are then entered in the receive GD packet in the higher priority class. When the program returns to the interrupted SFC, this is terminated immediately.</p>
<b>Data Consistency with GD</b>	<p>The following rules apply to the consistency of the data entered in the various memory areas.</p> <p>The following are consistent:</p> <ul style="list-style-type: none"><li>• The simple data types (bit, byte, word and double word)</li><li>• An array of the data types byte, word and double word up to a maximum length specific to the receiving CPU.</li></ul>
<b>Ensuring Consistency for an Entire GD Packet</b>	<p>A GD packet on a receiving CPU has a structure that does not automatically guarantee that its data originate from one and the same frame. This is, for example, the case when the packet consists of three GD elements.</p> <p>If, however, you require consistency for the entire GD packet, follow the procedure below in your program:</p> <ul style="list-style-type: none"><li>• Disable or delay the occurrence of higher priority interrupts and asynchronous errors by calling SFC39 “DIS_IRT” or SFC41 “DIS_AIRT”.</li><li>• Call SFC60 “GD_SND”.</li><li>• Enable the higher priority interrupts and asynchronous errors again by calling SFC40 “EN_IRT” or SFC42 “EN_AIRT”.</li></ul>

## Parameters

Table 16-3 Parameters for SFC61 “GD\_RCV”

Parameter	Declaration	Data Type	Memory Area	Description
CIRCLE_ID	INPUT	BYTE	I, Q, M, D, L, constant	Number of the GD circle into which the incoming GD packet will be entered. This number is specified during configuration of the global data with STEP 7. Permitted values: 1 to 16. The maximum number of possible GD circles can be found in the technical data for your CPU.
BLOCK_ID	INPUT	BYTE	I, Q, M, D, L, constant	Number of the GD packet in the selected GD circle in which the incoming data will be entered. This number is specified during configuration of the global data by STEP 7. Permitted values: 1 to 3. The maximum number of possible GD circles can be found in the technical data for your CPU.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information

## Error Information

Table 16-4 Specific Error Information for SFC61 “GD\_RCV”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8081	The GD packet selected with the parameters CIRCLE_ID and BLOCK_ID is not configured.
8082	Illegal value for the parameters CIRCLE_ID or BLOCK_ID or for both parameters.
8083	An error occurred during the execution of the SFC. The type of error is entered in the variable configured for the status information. This can be evaluated by your program.
8084	The execution of the SFC was terminated prematurely because SFC61 was called again for the same GD packet in a higher priority class (see “Interruptability”).
8085	An error occurred entering the status information in the configured variable.

### Note

Following each SFC60 call, you should evaluate the corresponding GD packet status and, if necessary, reset it.



# Communication SFBs for Configured Connections

# 17

## Chapter Overview

Section	Description	Page
17.1	Classification of the Communication SFBs for Configured Connections	17-2
17.2	Classification of the Parameters of Communication SFBs for Configured Connections	17-3
17.3	Uncoordinated Sending of Data with SFB8 "USEND"	17-7
17.4	Uncoordinated Receiving of Data with SFB9 "URCV"	17-8
17.5	Sending Segmented Data with SFB12 "BSEND"	17-10
17.6	Receiving Segmented Data with SFB13 "BRCV"	17-12
17.7	Reading Data from a Remote CPU with SFB14 "GET"	17-14
17.8	Writing Data to a Remote CPU with SFB15 "PUT"	17-16
17.9	Sending Data to a Printer With SFB 16 "PRINT"	17-18
17.10	Initiating a Complete Restart on a Remote Device with SFB19 "START"	17-25
17.11	Changing a Remote Device to the STOP State with SFB20 "STOP"	17-27
17.12	Initiating a Restart on a Remote Device with SFB21 "RESUME"	17-29
17.13	Querying the Status of a Remote Partner with SFB22 "STATUS"	17-31
17.14	Receiving the Status of a Remote Device with SFB23 "USTATUS"	17-33
17.15	Querying the Status of a CFB Instance with SFC62 "CONTROL"	17-35
17.16	Startup Routine of SFBs for Configured Connections	17-37
17.17	How Communication SFBs for Configured Connections React to Problems	17-39

## 17.1 Classification of the Communication SFBs for Configured Connections

### Classification

The communication SFBs for configured connections can be classified in two categories:

- Communication SFBs for data exchange
- communication SFBs for program management

### Communication SFBs for Data Exchange

Communication SFBs for data exchange are used to exchange data between two communication partners. If an SFB exists only on the local module this is referred to as unilateral data exchange, if an SFB exists on the local as well as on the remote module, this is referred to as a bilateral data exchange.

Table 17-1 Communication SFBs for Configured Connections for Data Exchange

SFB Name	Brief Description	Data Exchange
USEND/URCV	Uncoordinated data exchange using a send and a receive CFB	Bilateral
BSEND/BRCV	Exchange of blocks of data of up to 64 Kbytes in length <sup>1</sup> between a send CFB and a receive CFB	Bilateral
GET	Read data from a remote device	Unilateral
PUT	Write data to a remote device	Unilateral
PRINT	Send data, including formatting instructions to a printer	Unilateral

<sup>1</sup> In contrast to all other communication SFBs for configured connections, the maximum length of the blocks of data that are exchanged does not depend on the CPU used.

### Communication SFBs for Program Management

With communication SFBs for program management:

- You can control the operating mode of a remote device.
- You can receive information about the operating mode of a remote device.
- You can send messages to remote devices.

All communication SFBs for program management involve in unilateral data exchange except for SFB USTATUS.

With SFC62 "CONTROL" you query the status of the connection belonging to a communication SFB instance.

Table 17-2 Communication SFBs for Program Management

SFC/SFB Name	Brief Description
START	Initiate a complete restart in a remote device.
STOP	Initiate a STOP in a remote device.
RESUME	Initiate a restart in a remote device.
STATUS	Query the operating mode of a remote device.
USTATUS	Receive the status of a remote partner whenever its operating mode changes.
CONTROL	Query the connection status



## 17.2 Classification of the Parameters of Communication SFBs for Configured Connections

### Classification

The parameters of communication SFBs for configured connections can be divided into the following five categories according to their functions:

- Control parameters (used to activate a block).
- Addressing parameters (used to address the remote communication partner).
- Send parameters (point to the data areas that are to be sent to the remote partner).
- Receive parameters (point to the data areas where the data received from remote partners will be entered).
- Status parameters (used to monitor whether the block has completed its task without error or for the analysis of any errors that have occurred).

### Control Parameters

Data exchange will only be activated if the appropriate control parameters have a defined signal state when the SFB is called or when the signal state has undergone a specific change since the previous SFB call. Due to these two methods of activation, parameters are referred to as level-triggered and edge-triggered control parameters.

Table 17-3 Control Parameters for Communication SFBs for Configured Connections

Parameter	Meaning	Sender / Receiver	Activation of Function at the	Description
REQ	Request	Sender of job	Rising edge (compared to previous SFB call)	Activates the data exchange (if other conditions are satisfied)
R	Reset	Sender of job	Rising edge (compared to previous SFB call)	Aborts a currently active data exchange.
EN_R	Enabled to receive	Receiver of job	1 level	Signals "ready to receive".

## Addressing Parameters

Table 17-4 Addressing Parameters for Communication SFBs for Configured Connections

Parameter	Description	Take into Account
ID	Reference to the local connection description. The ID is assigned when the connection is configured.	<ul style="list-style-type: none"> <li>The ID must be specified in the form W#16#wxyz.</li> <li>In bilateral communication, the ID parameter exists in the SFB on the send side as well as in the SFB on the receive side.</li> <li>In unilateral communication no SFB exists on the remote module. You can choose between the following two procedures with respect to the connection description: <ul style="list-style-type: none"> <li>Storing the connection description on the local and remote partner. Advantage: The resources for the communication relation are permanently reserved with this configuration.</li> <li>Storing the connection description only on the local module.</li> </ul> </li> </ul>
R_ID	In bilateral communication you indicate that a send and a receive SFB belong together using the R_ID: The SFBs that belong together have the same value for R_ID.  This allows several SFB pairs to communicate on one logical connection.	<ul style="list-style-type: none"> <li>R_ID must be specified in the form DW#16#wxyzWXYZ.</li> <li>The block pairs of a logical connection specified by R_ID must be unique for this connection.</li> <li>R_ID has a different meaning in point-to-point communication using the CP 441.</li> </ul>
PI_NAME	With SFB 19 "START", SFB 20 "STOP" and SFB 21 "RESUME", pointer to the memory area in which the name of the program to be started is located (ASCII code).	In S7, all character strings are permitted that begin with P_PROGRAM.

### Note

The addressing parameters ID and R\_ID are evaluated only during the first call of the block (the actual parameters or the predefined values from the instance). The first call therefore specifies the communication relation (connection) with the remote partner until the next complete restart.

**Status Parameters**

With the status parameters, you monitor whether the block has completed its task correctly or whether it is still active. The status parameters also indicate errors.

**Note**

The status parameters are valid only for one cycle, in other words from the first instruction following the SFB call until the next SFB call. As a result, you must evaluate these parameters **every** time the block is completed.

Table 17-5 Status Parameters for Communication SFBs for Configured Connections

Parameter	Data Type	Sender / Receiver	Description
DONE	BOOL	Sender	0: Job was not yet started or is still active. 1: Job was executed without errors.
NDR	BOOL	Receiver	0: Job was not yet started or is still running. 1: Job was completed successfully.
ERROR	BOOL	Sender and receiver	ERROR Meaning 0 The value of STATUS is <ul style="list-style-type: none"> <li>• 0000H: neither warning nor error</li> <li>• unequal to 0000H: warning.</li> </ul> STATUS supplies detailed information.
STATUS	WORD	Sender and receiver	1 An error has occurred. STATUS supplies detailed information about the type of error.

**Send and Receive Parameters**

The SD\_i and ADDR\_i (with SFB15 “PUT”) send parameters and the RD\_i and ADDR\_i (with SFB14 “GET”) receive parameters are of the data type ANY, no bit fields and no variables of the type STRING, however, may be used.

If you do not use all send or receive parameters of an SFB, the first unused parameter must be a NIL pointer and the parameters used must be located one after the other and without any gaps.

During the first call, the connection and the maximum amount of data to be transferred on it per job are specified. A communication buffer is created internally in the system to ensure consistency of the data.

During the subsequent calls you can send/receive any amount of data, however, not more than during the first call.

The SFBs BSEND and BRCV are an exception to the rule. With them, you can transmit up to 64 Kbytes per job (see 17.5 and 17.6).

With communication SFBs for bilateral communication:

- The number of the SD\_i and RD\_i parameters used must match on the send and receive side.
  - The data types of the SD\_i and DR\_i parameters that belong together must match on the send and receive side.
  - The amount of data to be sent according to the SD\_i parameter must not exceed the area made available by the corresponding RD\_i parameter.
- ERROR = 1 and STATUS = 4 indicate that you have violated the above rules.

---

**Note**

The send and receive parameters are only evaluated when the block is first called (the actual parameters or the predefined values of the instance).

---

With the SFBs USEND, URCV, GET and PUT, the amount of data to be transmitted must not exceed a maximum user data length.

This maximum user data length depends on whether the remote partner is an S7-300 or an S7-400. This is shown in Table 17-6.

Table 17-6 Maximum User Data Length Depending on the Remote Partner

SFB	S7-300	S7-400
USEND/URCV	–	440
GET	210	450
PUT	164	404

**Data Consistency**

Access by the operating system to the data is packet-oriented. All the data of such a packet belong together and cannot be modified by other program sections during data transfer. This is known as data consistency.

In terms of consistency, the “weaker” communication partner determines the resulting length of consistent data for data transfer. The following are consistent:

- simple data types (bit, byte, word and double word)
- an array of data type byte up to a maximum length for the specific CPU (see /71/ , /101/ ).

## 17.3 Uncoordinated Sending of Data with SFB8 “USEND”

### Description

SFB8 “USEND” sends data to a remote partner SFB of the type “URCV” (the parameter R\_ID must be identical for both SFBs). The data is sent on a rising edge at control input REQ. The function is executed without coordination with the partner SFB. The data to be sent is referenced by the parameters SD\_1 to SD\_4 but not all four send parameters need to be used. You must, however, make sure that the areas defined by parameters SD\_i and RD\_i,  $1 \leq i \leq 4$  match each other in number, length and data type (RD\_i belongs to the corresponding partner SFB “URCV”). Successful completion of the transmission is indicated by the status parameter DONE having the value 1.

### Parameters

Table 17-7 Parameters for SFB8 “USEND”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Addressing parameter R_ID
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: data sent
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the first send data area
SD_2	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the second send data area
SD_3	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the third send data area
SD_4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the fourth send data area

Table 17-8 Specific Error Information for SFB8 “USEND”

### Error Information

ER-ROR	STATUS (decimal)	Explanation
0	11	Warning: new job cannot take effect since previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>connection description not loaded (local or remote)</li> <li>connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	4	Error in the send data area pointers SD_i involving data length or data type.
1	10	Access to the local user memory is not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>an instance DB was specified that does not belong to SFB8</li> <li>no instance DB was specified, but rather a shared DB.</li> <li>no instance DB found (loading new instance DB from PG).</li> </ul>
1	18	R_ID exists already in the connection ID.
1	20	Not enough work memory available.

## 17.4 Uncoordinated Receiving of Data with SFB9 “URCV”

### Description

SFB9 “URCV” receives data asynchronously from a remote partner SFB of the type “USEND” (the parameter R\_ID must be identical in both SFBs). If the value 1 is applied to the control input EN\_R when the block is called, the received data are copied to the configured receive areas. These data areas are referenced by the parameters RD\_1 to RD\_4. Remember that the area defined by the parameters RD\_i ( $1 \leq i \leq 4$ ) must match the areas of the corresponding SFB “USEND” defined with the parameters SD\_i ( $1 \leq i \leq 4$ ) in terms of number and data type. When the block is first called, the “receive mail box” is created. With all further calls, the data to be received must fit into this receive mail box. Completion of the copying job is indicated by the value 1 for the status parameter NDR.

### Parameters

Table 17-9 Parameters for SFB9 “URCV”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter enabled to receive
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Addressing parameter R_ID
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: new data accepted
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
RD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the first receive area
RD_2	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the second receive area
RD_3	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the third receive area
RD_4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the fourth receive area

Table 17-10 Specific Error Information for SFB9 "URCV"

**Error information**

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	9	Overrun warning: older received data are overwritten by newer received data.
0	11	Warning: the new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	4	Errors in the receive area pointers RD_i involving the data length or the data type.
1	9	Received data were overwritten.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the CFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB9</li> <li>• no instance DB was specified, but rather a shared DB.</li> </ul>
1	18	R_ID already exists in the connection ID.
1	19	The corresponding SFB "USEND" is sending data faster than it can be copied to the receive areas by SFB "URCV".
1	20	Not enough work memory available.

## 17.5 Sending Segmented Data with SFB12 “BSEND”

### Description

SFB12 “BSEND” sends data to a remote partner SFB of the type “BRCV” (the parameter R\_ID must be identical in the corresponding SFBs). With this type of data transfer, more data can be transported between the communications partners than is possible with all other communication SFBs for configured connections, namely up to 64 Kbytes (applies to all CPUs). The reason for this is that the data area to be transmitted is segmented. Each segment is sent individually to the partner and the sender waits for acknowledgement of reception before sending the next segment.

The send job is activated after calling the block and when there is a rising edge at the control input REQ. The start address of the data to be sent is specified by SD\_1, the length of the data field by LEN. The transmission of the data from the user memory is asynchronous to the processing of the user program. Successful completion of the transfer is indicated by the status parameter DONE having the value 1. If there is a rising edge at control input R, the current data transfer is canceled.

### Parameters

Table 17-11 Parameters for SFB12 “BSEND”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
R	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter reset: cancels the current job
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Addressing parameter R_ID. With a connection via the CP 441 to S5 or other devices, R_ID contains the address information of remote device. For further information, refer to the description of the CP 441.
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: the data transfer is completed.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the send area. The length information is not evaluated.
LEN	IN_OUT	WORD	I, Q, M, D, L	Length of the data field to be sent in bytes.



**Error Information**

Table 17-12 contains all the error information specific to SFB12 that can be output with the parameters ERROR and STATUS.

Table 17-12 Error Information for SFB12 "BSEND"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner SFB. The function cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	Error in the receive area pointer SD_1 involving the data length or the data type or the value 0 was transferred with LEN..
1	5	Reset request was executed.
1	6	Partner SFB is in the DISABLED state (EN_R has the value 0).
1	7	Partner SFB is in the wrong state (the receive block was not called again after the last data transmission).
1	8	Access to remote object in the user memory was rejected.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB12</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	18	R_ID already exists in the connection ID.
1	20	Not enough work memory available.

## 17.6 Receiving Segmented Data with SFB13 “BRCV”

### Description

SFB13 “BRCV” receives data from a remote partner SFB of the type “BSEND” (the parameter R\_ID must be identical in both SFBs). After it has been called and the value 1 is applied at the control input EN\_R, the block is ready to receive data. The start address of the receive area is specified by RD\_1.

Following each received data segment, an acknowledgement is sent to the partner SFB and the LEN parameter is updated. If the block is called during asynchronous reception of data, this leads to a warning being output in the STATUS status parameter; if the call is made when the value 0 is applied to control input EN\_R, reception is terminated and the SFB returns to its initial state. Error free reception of all the data segments is indicated by the status parameter NDR having the value 1. In this case, you must call the SFB with the value 0 at EN\_R again to make sure that the receive data area is not overwritten again before you have evaluated it.

### Parameters

Table 17-13 Parameters for SFB13 “BRCV”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter enabled to receive
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
R_ID	INPUT	DWORD	I, Q, M, D, L, constant	Addressing parameter R_ID. With a connection via the CP 441 to S5 or to other devices, R_ID contains the address information of the remote device. For further information, refer to the description of the CP 441.
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: new data accepted
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
RD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the receive area. The length information specifies the maximum length of the field to be received.
LEN	IN_OUT	WORD	I, Q, M, D, L	Length of the data received up to now in bytes

**Error Information**

Table 17-14 contains all the error information specific to SFB13 that can be output with the parameters ERROR and STATUS.

Table 17-14 Error Information for SFB13 "BRCV"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job is not effective since the previous job is not yet completed.
0	17	Warning: block receiving data asynchronously.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Function cannot be executed, for example, due to an error in the sequence.
1	4	Error in the receive area pointer RD_1 involving the data length or the data type (the block of data sent is longer than the receive area).
1	5	Reset request received, incomplete transfer.
1	8	Access to a remote object in the user memory was rejected.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB13</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	18	R_ID already exists in the connection ID.
1	20	Not enough work memory available.

## 17.7 Read Data from a Remote CPU with SFB14 “GET”

### Description

With SFB14 “GET” you can read data from a remote CPU.

With a rising edge at control input REQ, the relevant pointers to the areas to be read out (ADDR\_i) are sent to the partner CPU. The remote partner returns the data. The reply is evaluated to find out whether access problems occurred when reading the data and a data type check is made. If no errors occurred, the received data are copied to the configured receive areas (RD\_i) at the next SFB call. The completion of the job is indicated by a 1 at the status parameter NDR.

The read job can only be activated again after the previous job was completed.

Make sure that the areas defined with the parameters ADDR\_i and RD\_i ( $1 \leq i \leq 4$ ) match in terms of length and data type.

### Parameters

Table 17-15 Parameters for SFB14 “GET”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: data received from the partner CPU.
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointers to the areas on the partner CPU that will be read.
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	
RD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointers to the areas on the local CPU in which the data that have been read will be entered.
RD_2	IN_OUT	ANY	I, Q, M, D, T, C	
RD_3	IN_OUT	ANY	I, Q, M, D, T, C	
RD_4	IN_OUT	ANY	I, Q, M, D, T, C	

**Error Information**

Table 17-16 contains all the error information specific to SFB14 that can be output with the parameters ERROR and STATUS.

Table 17-16 Error Information for SFB14 "GET"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_i involving the data length or the data type.
1	8	Access error on the partner CPU
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB14</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.

## 17.8 Writing Data to a Remote CPU with SFB15 “PUT”

### Description

With SFB15 “PUT”, you can write data to a remote CPU.

With a rising edge at the control input REQ, the pointers to the areas to be written (ADDR\_i) and the data (SD\_i) are sent to the partner CPU. The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement. The execution acknowledgement is evaluated. If no errors occur, this is indicated at the next SFB call with the status parameter DONE having the value 1.

The write job can only be activated again after the last job was completed.

Make sure that the areas defined with the parameters ADDR\_i and RD\_i ( $1 \leq i \leq 4$ ) match in terms of number, length, and data type.

### Parameters

Table 17-17 Parameters for SFB15 “PUT”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: function executed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
ADDR_1	IN_OUT	ANY	e.g. I, Q, M, D	Pointers to the areas on the partner CPU in which the data will be written.
ADDR_2	IN_OUT	ANY	e.g. I, Q, M, D	
ADDR_3	IN_OUT	ANY	e.g. I, Q, M, D	
ADDR_4	IN_OUT	ANY	e.g. I, Q, M, D	
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointers to the areas on the local CPU containing the data to be sent.
SD_2	IN_OUT	ANY	I, Q, M, D, T, C	
SD_3	IN_OUT	ANY	I, Q, M, D, T, C	
SD_4	IN_OUT	ANY	I, Q, M, D, T, C	

**Error Information**

Table 17-18 contains all the error information specific to SFB15 that can be output with the parameters ERROR and STATUS.

Table 17-18 Error Information for SFB15 "PUT"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_i involving the data length or the data type.
1	8	Access error on the partner CPU
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB15</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.

## 17.9 Sending Data to a Printer With SFB 16 "PRINT"

### Description

SFB 16 "PRINT" sends data and a formatting instruction to a remote printer.

When there is a rising edge at control input REQ, the format description (FORMAT) and the data (SD\_i) are sent to the printer selected with ID and PRN\_NR.

If you do not use all four send areas, you must make sure that the first area is described by the SD\_1 parameter, the second area (if it exists) by the SD\_2 parameter, the third area (if it exists) by SD\_3.

Successful execution of the job is indicated by the DONE status parameter, errors are indicated by the ERROR and STATUS status parameters.

### Parameters

Table 17-19 Parameters for SFB 16 "PRINT"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Request control parameter
ID	INPUT	WORD	I, Q, M, D, L, constant	ID addressing parameter
DONE	OUTPUT	BOOL	I, Q, M, D, L	DONE status parameter: Send job completed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	ERROR status parameter
STATUS	OUTPUT	WORD	I, Q, M, D, L	STATUS status parameter
PRN_NR	IN_OUT	BYTE	I, Q, M, D, L	Printer number
FORMAT	IN_OUT	STRING	I, Q, M, D, L	Format description
SD_1	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the first send area
SD_2	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the second send area
SD_3	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the third send area
SD_4	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the fourth send area



**In\_Out Parameter  
FORMAT**

The FORMAT character string contains printable characters and format elements. It has the following structure:

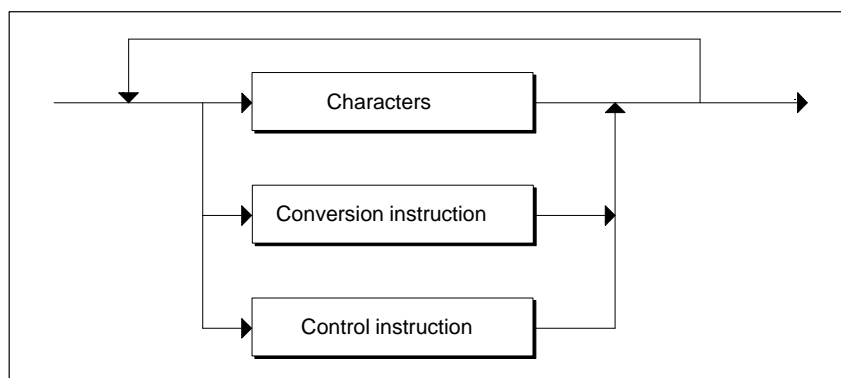


Figure 17-1 Syntax Diagram of the FORMAT Character String

For each send area to be printed (SD\_1 to SD\_4) there must be one conversion instruction in FORMAT. The conversion instructions are applied to the send areas (SD\_i) in the order in which they are formulated. Characters and instructions can follow each other in any order.

- Characters

The following characters are permitted

- All printable characters
- \$\$ (Dollar character), \$' (single inverted comma), \$L and \$l (line feed), \$P and \$p (page), \$R and \$r (carriage return), \$T and \$t (tabulator)

- Conversion Instruction

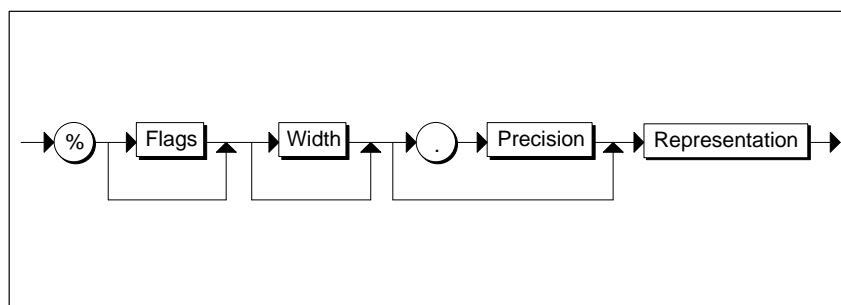


Figure 17-2 Syntax Diagram of a Conversion Instruction

Table 17-20 The Elements: Flags, Width, Precision and Representation of a Conversion Instruction

Element of a Conversion Instruction	Meaning
Flags	<ul style="list-style-type: none"> <li>• None: right-justified output</li> <li>• – : left-justified output</li> </ul>
Width	<ul style="list-style-type: none"> <li>• None: output in standard representation</li> <li>• n: exactly n characters are output. If the output is right-justified, this may be preceded by blanks, with left-justified output the blanks come after the characters.</li> </ul>
Precision	<p>The precision is only relevant for representations A, D, F and R (see Table 17-21).</p> <ul style="list-style-type: none"> <li>• None: output in standard representation</li> <li>• 0: no output of the decimal point or decimal places in the F and R representations</li> <li>• n: <ul style="list-style-type: none"> <li>– with F and R: output of the decimal point and n decimal places</li> <li>– with A and D (date): number of digits for the year: possible values 2 and 4.</li> </ul> </li> </ul>
Representation	<p>Table 17-21 contains the following:</p> <ul style="list-style-type: none"> <li>• The possible representations</li> <li>• The data types possible for each representation</li> <li>• The standard format for each representation (the printout is in the standard representation if no width and no precision are specified in the FORMAT parameter) and their maximum length</li> </ul>

Table 17-21 Possible Representations in the Conversion Instruction of the FORMAT Parameter

Representation	Possible Data Types	Standard Representation		Comments
		Example	Length	
A, a	DATE	25.07.1996	10	–
	DWORD			
C, c	CHAR	K	1	–
	BYTE	M	1	
	WORD	KL	2	
	DWORD	KLMN	4	
	ARRAY of CHAR	KLMNOP	Number of characters	
	ARRAY of BYTE			
D, d	DATE	1996–07–25	10	–
	DWORD			
F, f	REAL	0.345678	8	–
	DWORD			
H, h	All data types incl. ARRAY of BYTE	Depending on data type	Depending on data type	Hexadecimal representation
I, i	INT	– 32 768	max. 6	–
	WORD	– 2 147 483 648	max. 11	
N, n	WORD	Text output	–	The corresponding send area SD_i contains a reference (number) to a text to be printed. The text is on the module (e.g. CP 441) that creates a printable string. If no text is found under the specified number, ***** is output.
R, r	REAL	0.12E–04	8	–
	DWORD			
S, s	STRING	Text output		–
T, t	TIME	2d_3h_10m_5s_250ms	max. 21	If an error occurs, ***** is output.
	DWORD			
U, u	BYTE	255	max. 3	–
	WORD	65 535	max. 5	
	DWORD	4 294 967 295	max. 10	
X, x	BOOL	1	1	–
	BYTE	101 ..	8	
	WORD	101 ..	16	
	DWORD	101 ..	32	
Z, z	TIME_OF_DAY	15:38:59.874	12	–

At the points in Table 17-21 at which a maximum length is specified for the representation, the actual length can of course be shorter.

**Note**

With the data types C and S, the following points depend on the printer being used:

- which characters can be printed
- what the printer prints for non-printable characters, unless the printer driver has a conversion table for these characters.

- Control Instruction

Using the control instruction you can do the following:

- Print the characters % and \
- Change printer settings

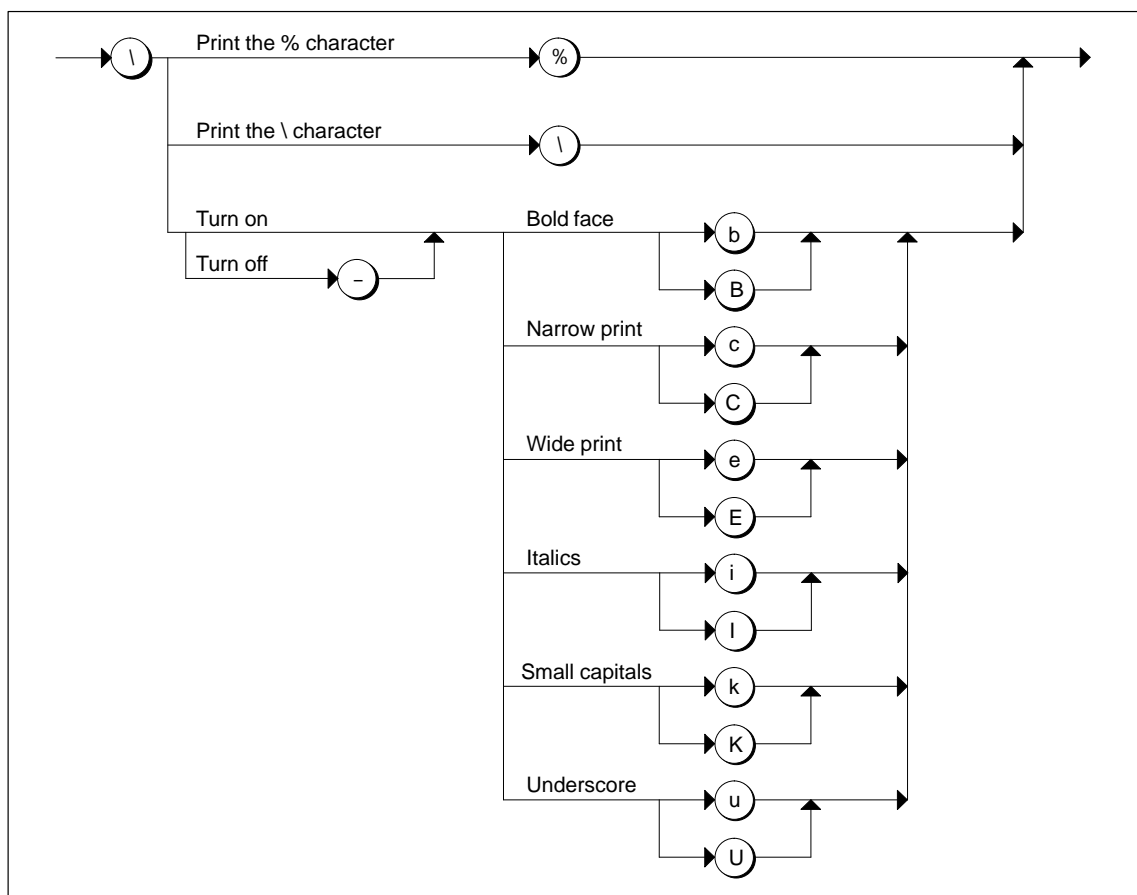


Figure 17-3

Syntax Diagram of the Control Instruction

If you attempt to disable, for example, a font that is not enabled or a function that the printer does not recognize the control instruction is ignored.

Table 17-22 Errors Occurring With the In/Out Parameter FORMAT

Error	Printer Output
Conversion instruction cannot be executed	* characters are output according to the (maximum) length of the default representation or the specified width.
Specified width too small	In the representations A, C, D, N, S, T, and Z, as many characters are printed as specified by the selected width. With all other representations, * characters are printed across the specified width.
Too many conversion instructions	The conversion instructions for which there is no send area pointer SD_i are ignored.
Too few conversion instructions	Send areas for which there is no conversion instruction are not printed out.
Undefined or unsupported conversion instructions	***** is printed out.
Incomplete conversion instruction	***** is printed out.
Undefined or unsupported control instructions	Control instructions that do not comply with the Syntax shown in Figure 17-3 are ignored.

**Error Information**

Table 17-23 contains all the error information specific to SFB 16 “PRINT” that can be printed out using the ERROR and STATUS parameters.

Table 17-23 Error Information for SFB 16 “PRINT”

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is not yet completed.
1	1	Communication problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgment from printer. The function cannot be executed.
1	3	PRN_NR is unknown on the communication link specified by the ID.
1	4	Error in the FORMAT in/out parameter or in the send area pointers SD_i in terms of the data length or data type.
1	6	The remote printer is OFFLINE.
1	7	The remote printer is not in the correct status (for example paper out).
1	10	Access to the local user memory not possible (for example access to a deleted DB).
1	13	Error in the FORMAT in/out parameter
1	20	Not enough work memory

**Number of  
Transferrable Data**

The amount of data that can be transferred to a remote printer must not exceed a maximum length.

This maximum data length is calculated as follows:

$$\text{maxleng} = 420 - \text{format}$$

Format is the current length of the **FORMAT** parameter in bytes. The data to be printed can be distributed on one or more send areas.

## 17.10 Initiating a Complete Restart on a Remote Device with SFB 19 "START"

### Description

If there is a rising edge at control input REQ, SFB19 "START", this activates a complete restart on the remote device addressed by the ID. The following conditions must be met if the remote device is a CPU:

- The CPU must be in the STOP status.
- The keyswitch of the CPU must be set to "RUN" or "RUN-P".

Once the complete restart is completed, the device changes to the RUN mode and sends a positive execution acknowledgement. When the positive acknowledgement is evaluated, the status parameter DONE is set to 1. If any errors occur, they are indicated by the status parameters ERROR and STATUS.

A further complete restart can only be activated in the same remote device after the last complete restart is completed.

### Parameters

Table 17-24 Parameters for SFB19 "START"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: function executed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
PI_NAME	IN_OUT	ANY	I, Q, M, D, T, C	Pointer to the memory area in which the name of the program (ASCII code) to be started is located. With an S7 PLC, this name must be P_PROGRAM.
ARG	IN_OUT	ANY	I, Q, M, D, T, C	Execution argument. Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 programmable controller.
IO_STATE	IN_OUT	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 programmable controller.

**Error Information**

Table 17-25 contains all the error information specific to SFB19 that can be output with the parameters ERROR and STATUS.

Table 17-25 Error Information for SFB19 "START"

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered for PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG involving the data length or the data type.
1	7	No complete restart possible on the partner device.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB19</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.



## 17.11 Changing a Remote Device to the STOP State with SFB20 “STOP”

### Description

If there is a rising edge at control input REQ, SFB20 “STOP”, this activates a change to the STOP state on the remote device addressed by the ID. The mode change is possible when the device is in the RUN, HOLD or start-up modes.

Successful execution of the job is indicated at the status parameter DONE. If any errors occur, they are indicated in the status parameters ERROR and STATUS.

The mode change can only be started again in the same remote device when the previous SFB20 call has been completed.

### Parameters

Table 17-26 Parameters for SFB20 “STOP”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: function executed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
PI_NAME	IN_OUT	ANY	I, Q, M, D	Pointer to memory area in which the name of the program (ASCII code) to be started is located. With an S7 PLC, this name must be P_PROGRAM.
IO_STATE	IN_OUT	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 programmable controller.

**Error Information**

Table 17-25 contains all the error information specific to SFB20 that can be output with the parameters ERROR and STATUS.

Table 17-27 Error Information for SFB20 "STOP"

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered for PI_NAME is unknown.
1	4	Error in the pointer PI_NAME involving the data length or the data type.
1	7	The partner device is already in the STOP state.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB20</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.

## 17.12 Initiating a Restart on a Remote Device with SFB21 “RESUME”

### Description

If there is a rising edge at control input REQ, SFB21 “RESUME” activates a restart on the remote device selected with the ID.

The following conditions must be met if the remote device is a CPU:

- The CPU must be in the STOP mode.
- The keyswitch of the CPU must be set to “RUN” or “RUN-P”.
- When you created the configuration with STEP 7, you allowed for a manual restart.
- There must be no condition preventing a restart.

Once the restart has been completed, the device changes to the RUN mode and sends a positive execution acknowledgement. When the positive acknowledgement is evaluated, the status parameter DONE is set to 1. Any errors that occur are indicated in the status parameters ERROR and STATUS.

A restart can only be activated again in the same remote device after the previous restart has been completed.

### Parameters

Table 17-28 Parameter for SFB21 “RESUME”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
DONE	OUTPUT	BOOL	I, Q, M, D, L	Status parameter DONE: function executed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
PI_NAME	IN_OUT	ANY	I, Q, M, D	Pointer to memory area in which the name of the program (ASCII code) to be started is located. With an S7 PLC, this name must be P_PROGRAM.
ARG	IN_OUT	ANY	I, Q, M, D, T, C	Execution argument. Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 programmable controller.
IO_STATE	IN_OUT	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 programmable controller.

**Error Information**

Table 17-29 contains all the error information specific to SFB21 that can be output with the parameters ERROR and STATUS.

Table 17-29 Error Information for SFB21 "RESUME"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>• connection description not loaded (local or remote)</li> <li>• connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG involving the data length or the data type.
1	7	Restart not possible
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>• an instance DB was specified that does not belong to SFB21</li> <li>• no instance DB was specified, but rather a shared DB.</li> <li>• no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.

## 17.13 Querying the Status of a Remote Partner with SFB22 “STATUS”

### Description

Using SFB22 “STATUS”, you can query the status of a remote communications partner.

If there is a rising edge at control input REQ, a job is sent to the remote partner. The reply is evaluated to determine whether problems have occurred. If no errors occurred, the received status is copied to the variables PHYS, LOG and LOCAL with the next SFB call. The completion of this job is indicated by the status parameter NDR having the value 1.

You can only query the status of the same communications partner again after the last query is completed.

### Parameters

Table 17-30 Parameters for SFB22 “STATUS”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter request
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: new device status received
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
PHYS	IN_OUT	ANY	I, Q, M, D	Physical status (minimum length: one byte), possible values: <ul style="list-style-type: none"> <li>10H functioning</li> <li>13H service required</li> </ul>
LOG	IN_OUT	ANY	I, Q, M, D	Logical status (minimum length: one byte), possible value: <ul style="list-style-type: none"> <li>00H status change permitted</li> </ul>
LOCAL	IN_OUT	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length: one byte)

**In/Out Parameter  
LOCAL**

If the communications partner is an S7 CPU, the in/out parameter LOCAL contains its current status. The first byte is reserved, the second byte contains an ID for the status.

Table 17-31 Relationship between Statuses and Identifiers

Status	Corresponding Identifier
STOP	00H
Complete restart	01H
RUN	02H
Restart	03H
HOLD	04H
DEFECTIVE	05H

**Error Information**

Table 17-32 contains all the error information specific to SFB22 that can be output with the parameters ERROR and STATUS.

Table 17-32 Error Information for SFB22 "STATUS"

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job is not effective since the previous job is not yet completed.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>connection description not loaded (local or remote)</li> <li>connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Error in PHYS, LOG or LOCAL involving the data length or data type.
1	8	Access to a remote object was rejected.
1	10	Access to a local user memory not possible (for example, access to a deleted DB).
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>an instance DB was specified that does not belong to SFB22</li> <li>no instance DB was specified, but rather a shared DB.</li> <li>no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	20	Not enough work memory available.

## 17.14 Receiving the Status of a Remote Device with SFB23 “USTATUS”

### Description

SFB23 “USTATUS” receives the device status of a remote communication partner. The partner sends its status unsolicited when a change occurs if this is configured in STEP 7.

If the value 1 is applied to the control input EN\_R when the CFB is called and there is a frame from the partner, the status information is entered in the variables PHYS, LOG and LOCAL the next time the SFB is called. Completion of this job is indicated by the status parameter NDR having the value 1.

The transfer of the device status must be enabled on the connection used by USTATUS.

### Note

You can only use one instance of SFB23 per connection.

### Parameters

Table 17-33 Parameters for SFB23 “USTATUS”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter enabled to receive
ID	INPUT	WORD	I, Q, M, D, L, constant	Addressing parameter ID
NDR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter NDR: new device status accepted
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
PHYS	IN_OUT	ANY	I, Q, M, D	Physical status (minimum length one byte), possible values: <ul style="list-style-type: none"> <li>• 10H functional</li> <li>• 13H service required</li> </ul>
LOG	IN_OUT	ANY	I, Q, M, D	Logical status (minimum length one byte), possible value: <ul style="list-style-type: none"> <li>• 00H status change permitted</li> </ul>
LOCAL	IN_OUT	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length one byte)

**In/Out Parameter  
LOCAL**

If the communications partner is an S7 CPU, the in/out parameter LOCAL contains its current status. The first byte is reserved, the second byte contains an ID for the status.

Table 17-34 Relationship between Statuses and Identifiers

Status	Corresponding Identifier
STOP	00H
Complete restart	01H
RUN	02H
Restart	03H
HOLD	04H
DEFECTIVE	05H

**Error Information**

Table 17-35 contains all the error information specific to SFB23 that can be output with the parameters ERROR and STATUS.

Table 17-35 Error Information for SFB23 "USTATUS"

ERROR	STATUS (decimal)	Explanation
0	9	Overflow warning: an older device status has been overwritten by a more recent device status.
1	1	Communications problems, for example <ul style="list-style-type: none"> <li>connection description not loaded (local or remote)</li> <li>connection interrupted (e.g. cable, CPU off, CP in STOP mode)</li> </ul>
1	4	Error in PHYS, LOG or LOCAL involving the data length or data type.
1	9	Received data were overwritten.
1	10	Access to a local user memory not possible (for example, access to a deleted DB).
1	12	When the SFB was called, <ul style="list-style-type: none"> <li>an instance DB was specified that does not belong to SFB23</li> <li>no instance DB was specified, but rather a shared DB.</li> <li>no instance DB found (loading a new instance DB from the PG).</li> </ul>
1	18	There is already an instance for SFB23 "USTATUS" for the connection identified by ID.
1	19	The remote CPU sends a data faster than it can be accepted in the user program by the SFB.
1	20	Not enough work memory available.



## 17.15 Querying the Status of the Connection Belonging to a Communication SFB Instance with SFC62 “CONTROL”

### Description

With SFC62 “CONTROL”, you can query the status of a connection belonging to a local communication SFB instance.

After calling the system function with the value 1 at control input EN\_R, the current status of the connection belonging to the communication SFB instance selected with I\_DB is queried.

### Parameters

Table 17-36 Parameters for SFC62 “CONTROL”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter enabled to receive
I_DB	INPUT	BLOCK_DB	I, Q, M, D, L, constant	Number of the instance DB
OFFSET	INPUT	WORD	I, Q, M, D, L, constant	Number of the data record in the multiple instance DB (if no multiple instance DB exists, 0 must be entered here).
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Error information
ERROR	OUTPUT	BOOL	I, Q, M, D, L	Status parameter ERROR
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS
I_TYP	OUTPUT	BYTE	I, Q, M, D, L	Identifier for the block type belonging to the selected instance
I_STATE	OUTPUT	BYTE	I, Q, M, D, L	Not relevant for the user. Reserved for internal diagnostics.
I_CONN	OUTPUT	BOOL	I, Q, M, D, L	Status of the corresponding connection, possible values: <ul style="list-style-type: none"> <li>0: Connection terminated or not established</li> <li>1: Connection exists</li> </ul>
I_STATUS	OUTPUT	WORD	I, Q, M, D, L	Status parameter STATUS of the queried communication SFB instance

**Output Parameter  
I\_TYP**

Table 17-37 lists the different SFB types and the corresponding identifiers.

Table 17-37 Identifiers of the SFB Types

SFB Type	Identifier (W#16#...)
USEND	00
URCV	01
BSEND	04
BRCV	05
GET	06
PUT	07
PRINT	08
START	0B
STOP	0C
RESUME	0D
STATUS	0E
USTATUS	0F
ALARM	15
ALARM_8	16
ALARM_8P	17
NOTIFY	18
AR_SEND	19
(no SFB exists; I_DB or OFFSET wrong)	FF

**Error Information**

The output parameter RET\_VAL can have the following two values with SFC62 “CONTROL”:

- 0000H: no error occurred during execution of the SFC.
- 8000H: an error occurred during execution of the SFC.

**Note**

Even if the value 0000H is indicated in the output parameter RET\_VAL, the output parameters ERROR and STATUS should be evaluated.

Table 17-38 Specific Error Information for SFC62 “CONTROL”

ERROR	STATUS (decimal)	Explanation
1	10	Access to local user memory is not possible (for example, a memory byte was specified as the actual parameter for I_TYP and this memory byte does not exist in the CPU being used).
1	12	For the number specified with I_DB, <ul style="list-style-type: none"> <li>• there is no instance DB, but rather a shared DB,</li> <li>• there is no DB, or the instance has been destroyed.</li> </ul>

## 17.16 Startup Routine of Communication SFBs for Configured Connections

### Precondition

In the following description it is assumed that:

- The connection descriptions (SDBs) exist on the modules.
- The configured connections have been established.
- The actual parameter for the ID matches the configured connection ID for each SFB.

### Reaction to a Complete Restart

During a complete restart all communication SFBs are set to the NO\_INIT status. The actual parameters stored in the instance DBs are not changed.

### Complete Restart with CFBs for Bilateral Data Exchange

In general, the two modules with communication SFBs for bilateral data exchange do not both go through a complete restart simultaneously. The reaction of the SFB is governed by the rules below:

The receive blocks (SFBs URCV, BRCV) react as following:

- If the SFB has received a job but has not acknowledged this job at the time of the complete restart, it generates a sequence abort frame (CFB, BRCV) and then immediately branches to the NO\_INIT status.
- With SFB BRCV, it is possible that another data segment will be received despite having sent the sequence abort. This will be discarded locally.
- SFB URCV changes to the NO\_INIT status immediately.

The send blocks (SFBs USEND, BSEND) react as follows:

- If SFB BSEND has started a job sequence that has not yet been completed, it sends a sequence abort when the complete restart is initiated. It then branches to the NO\_INIT status immediately afterwards. An acknowledgement that arrives at a later time is discarded locally.
- If SFB BSEND has already sent or received a sequence abort when the complete restart is requested, it changes immediately to the NO\_INIT status.
- In all other cases and whenever the SFB sends only messages (for example SFB USEND), local processing is aborted and the SFB immediately branches to the NO\_INIT status.

**Complete Restart  
with  
Communication  
SFBs for Unilateral  
Data Exchange**

It is assumed that the server on the communication partner is operational after the connections have been established, in other words that it can process jobs or output messages at any time.

Communication SFBs that send out jobs and expect acknowledgements react to a complete restart as follows:

The current processing is aborted and the CFB branches to the NO\_INIT status immediately afterwards. If an acknowledgement for the job sent prior to the complete restart arrives later, it is discarded locally.

A new job may have been sent before the acknowledgement of the earlier job is received.

Communication SFBs that output or receive messages react as follows:

- The current processing is aborted and the CFB branches to the NO\_INIT status immediately afterwards.
- With SFB USTATUS, messages that arrive during the NO\_INIT and DISABLED statuses are discarded locally.

**Reaction to a  
Restart**

The communication SFBs for configured connections are set to the NO\_INIT status only during a complete restart. This means that they react like user function blocks that can be resumed following a restart.

**Reaction to a  
Memory Reset**

A memory reset always causes all connections to be terminated. Since a complete restart is the only possible startup type for the user program after a memory reset, all communication SFBs for configured connections (if they still exist) are set to the NO\_INIT status and initialized. Partner blocks in a module whose memory was not reset change to the IDLE, ENABLED or DISABLED statuses as a reaction to the connection being terminated.

## 17.17 How CFBs React to Problems

### Connection Terminated

The connections allocated to the communication SFB instances are monitored.

If a connection is terminated, the reaction of the communication SFB depends on its internal status.

If the break down of the connection is detected while the block is in the IDLE or ENABLED status, the SFB reacts as follows:

- It branches to the ERROR status and outputs the error ID “Communication problems” at the ERROR and STATUS output parameters.
- When it is next called, the block returns to its original status and checks the connection again.

A communication SFB that is not in the IDLE or DISABLED statuses reacts as follows:

- It aborts processing, changes to the ERROR status immediately or at the next block call and outputs the error ID “Communication problems” at the ERROR and STATUS output parameters.
- When it is next called, the block changes to the IDLE, DISABLED or ENABLED status. In the IDLE and ENABLED status the connection is checked again.

This procedure will also be executed if the connection has again been set up in the meantime.

### Power Down

A power down with battery backup followed by a restart causes all established connections to be terminated. The points made above therefore apply to all blocks involved.

If there is a power down with battery backup followed by an automatic complete restart, the points made about terminated connections and complete restarts apply.

In the special case of an automatic complete restart without battery backup, where a memory reset is executed automatically after power returns, the communication SFBs for configured connections react as described in the section “*Startup Routine of the Communication SFBs For Configured Connections*”.

### Reaction to Operating Mode Changes

If the operating mode changes between the STOP, START, RUN, and HOLD statuses, the communication SFB remains in its current status (exception: during a complete restart, it changes to the NO\_INIT status). This applies both to SFBs for unilateral as well as SFBs for bilateral communication.

### **Error Interface to the User Program**

If an error occurs during the processing of a communication SFB, it always changes to the the ERROR status. At the same time the ERROR output parameter is set to 1 and the corresponding error ID is entered in the STATUS output parameter. You can evaluate this error information in your program.

Examples of possible errors:

- Error when collecting send data.
- Error when copying receive data into the receive areas (for example attempting to access a DB that does not exist).
- The length of the data area sent does not match the length of the receive area specified in the partner SFB.

# Communication SFCs for Non-Configured Connections 18

## Chapter Overview

Section	Description	Page
18.1	Differences Compared with Communication SFBs for Configured Connections	18-2
18.2	Overview	18-4
18.3	Common Parameters of the Communication SFCs	18-7
18.4	Data Consistency with GET and PUT SFCs	18-8
18.5	Sending Data to a Communication Partner outside the Local S7 Station with SFC65 "X_SEND"	18-11
18.6	Receiving Data from a Communication Partner outside the Local S7 Station with SFC66 "X_RCV"	18-13
18.7	Reading Data from a Communication Partner outside the Local S7 Station with SFC67 "X_GET"	18-17
18.8	Writing Data to a Communication Partner outside the Local S7 Station with SFC68 "X_PUT"	18-19
18.9	Aborting an Existing Connection to a Communication Partner outside the Local S7 Station with SFC69 "X_ABORT"	18-21
18.10	Reading Data from a Communication Partner within the Local S7 Station with SFC72 "I_GET"	18-22
18.11	Writing Data to a Communication Partner within the Local S7 Station with SFC73 "I_PUT"	18-24
18.12	Aborting an Existing Connection to a Communication Partner within the Local S7 Station with SFC74 "I_ABORT"	18-26
18.13	Error Information of the Communication SFCs for Non-Configured Connections	18-27

## 18.1 Differences Compared with Communication SFBs for Configured Connections

### Selecting Methods

Apart from global data communication, there are two other methods of exchanging data between CPUs/FMs of SIMATIC S7 programmable controllers:

- Data exchange using communication SFCs for non-configured connections
- Data exchange using communication SFBs for configured connections

Which method you choose, depends on the SIMATIC S7 programmable controller you are using (S7-300, S7-400) and on other parameters for data exchange. The following table contains a list of criteria on which you can base your selection.

Table 18-1 Comparison of Communication SFCs for Non-Configured Connections and Communication SFBs on Configured Connections

Criterion	Communication SFCs for Non-Configured Connections	Communication SFBs for Configured Connections
Availability of the blocks	S7-300 and S7-400	S7-400 only
Connections	The connection is not configured but is established while the SFC is being executed. Depending on the parameter assignment, the connection either remains established after the job is completed or it is terminated. If a connection cannot be established temporarily, the corresponding job cannot be sent.	Connections are configured during system configuration.
Change to the STOP mode	If the CPU that initiated a job (and therefore established the connection) changes to STOP during a data transfer, all the connections it established are terminated.	The connection is maintained in the STOP mode.
More than one connection to a partner	At any one time, a maximum of one connection is possible to a communication partner.	You can establish several connections to the same partner.
Address range	Modules can be addressed in the local station or in the MPI subnet	Modules can be addressed on the MPI network, on PROFIBUS or on Industrial Ethernet



Table 18-1 Comparison of Communication SFCs for Non-Configured Connections and Communication SFBs on Configured Connections, continued

Criterion	Communication SFCs for Non-Configured Connections	Communication SFBs for Configured Connections
Number of communication partners	The number of communication partners that can be reached one after the other is not restricted by the connection resources available (see /70/, /101/). (The connections can be established and terminated again while the program is running.)	The number of simultaneously obtainable communication partners is restricted to the number of connection resources available. It also depends on the CPU being used (see /70/, /101/).
Maximum user data length	A user data length of 76 bytes is guaranteed.	The maximum transferrable user data length depends on the block type (USEND / URCV, GET, etc.) and on the communication partner (S7-300 or S7-400).
Number of variables transferred per call	You can only transfer one variable.	You can transfer a maximum of four variables.
Classification of the blocks	The communication SFCs for non-configured connections are system functions and do not require user memory.	The communication SFBs for configured connections are system function blocks and require an instance DB for the current parameters and static data.
Dynamic modification of the address parameters	Dynamic modification of the address parameters is possible: on completion of the active job, you can address other communication partners.	Dynamic modification of address parameters is not possible: the connection is specified and fixed by the first block call and remains unchanged until the next complete restart.

## 18.2 Overview

### Classification of the SFCs

With the communication SFCs for non-configured connections, you can exchange data between an S7 CPU and other modules with communication functionality, if the communication partners are connected to the same MPI subnet or if they belong to the same S7 station. Communication with stations in other subnets is not possible with the communication SFCs for non-configured connections.

The communication SFCs for non-configured connections can be run on all CPUs of the S7-300 and S7-400. With these CPUs, you can also write variables to the CPUs of the S7-200 and read variables from them.

if the communication partner does not belong to the same S7 station, but it is attached to the same MPI subnet, data is exchanged using one of the blocks listed in Table 18-2.

Table 18-2 Blocks for Communication between S7 Stations

Block	Brief Description
SFC65 "X_SEND"/ SFC66 "X_RCV"	Data exchange between communication partners using a send and a receive SFC
SFC67 "X_GET"	Read a variable from a communication partner
SFC68 "X_PUT"	Write a variable to a communication partner
SFC69 "X_ABORT"	Abort an existing connection to a communication partner. Resources at both ends are released.

If both communication partners belong to the same S7 station, data is exchanged using one of the blocks listed in Table 18-3.

Table 18-3 Blocks for Communication within the Same S7 Station

Block	Brief Description
SFC72 "I_GET"	Read a variable from a communication partner (for example FM)
SFC73 "I_PUT"	Write a variable to a communication partner (for example FM)
SFC74 "I_ABORT"	Abort an existing connection to a communication partner. Resources at both ends are released.

### Maximum User Data Length

The communication SFCs for non-configured connections are integrated on all CPUs of the S7-300 and S7-400.

It is guaranteed that 76 bytes of user data can be transferred with all SFCs (parameter SD or RD).

## Connection to the Communication Partner

With the communication SFCs for non-configured connections, the connection is established while the SFC is being executed. Depending on the value you assign to the CONT input parameter, the connection either remains established or is terminated on completion of the data exchange. This means that the communication has the following characteristics:

- The number of communication partners that can be reached one after the other is higher than the number of communication partners that can be reached simultaneously (the number depends on the specific CPU, see /70/, /101/).
- If no connection can currently be established to a communication partner because the connection resources (on the local CPU or on the communication partner) are all being used, this is indicated in RET\_VAL. You must then trigger the job again later at a suitable point in time. There is, however, no guarantee that later connection establishment will be successful. If necessary, check the use of connection resources in your program and use a CPU with more resources.

Existing connections of communication SFBs for configured connections cannot be used by the communication SFCs for non-configured connections.

Once you have triggered a job, the connection established for the job can only be used for this particular job. Other jobs involving the same communication partner can then only be executed after the current job is completed.

### Note

If your program includes several jobs involving the same communication partner, make sure that you call the SFCs for which W#16#80C0 is entered in RET\_VAL again later at a suitable point in time.

## Identifying a Job

If you have triggered a data transfer or a connection abort with one of the communication SFCs for non-configured connections and you call this SFC again before the current transfer is completed, the reaction of the SFC depends on whether the new call involves the same job. Table 18-4 explains which input parameters specify a job for every SFC. If the parameters match those of a job that is not yet completed, the SFC call counts as a follow-on call.

Table 18-4 Input Parameters that Identify a Job

SFC	Job identified by
65 "X_SEND"	DEST_ID, REQ_ID
67 "X_GET"	DEST_ID, VAR_ADDR
68 "X_PUT"	DEST_ID, VAR_ADDR
69 "X_ABORT"	DEST_ID
72 "I_GET"	IOID, LADDR, VAR_ADDR

Table 18-4 Input Parameters that Identify a Job, continued

SFC	Job identified by
73 "I_PUT"	IOID, LADDR, VAR_ADDR
74 "I_ABORT"	IOID, LADDR

**Reaction to Interrupts**

The communication SFCs for non-configured connections can be interrupted by higher priority OBs. If the same SFC with the identical job is called again by the interrupting OB, this second call is aborted and a corresponding entry made in RET\_VAL. The execution of the interrupted SFC is then continued.

**Access to the Work Memory of the CPU**

Regardless of the number of user data to be transferred, the communication functions of the operating system access the work memory of the CPU in fields of the maximum length, so that the interrupt reaction time is not extended by the use of communication functions.

Depending on how you set the maximum cycle load resulting from communication with STEP 7, the work memory can be accessed several times during the execution of a job by the communication functions of the operating system.

**Client Changes to STOP**

If the CPU that initiated a job (and therefore established the connection) changes to STOP during a data transfer, all the connections it established are terminated.

**Making Program Changes**

All parts of your program that immediately affect the calls for communication SFCs for non-configured connections must only be modified in the STOP mode. This includes, in particular, deleting FCs, FBs, or OBs containing calls for communication SFCs for non-configured connections.

After modifying the program, you must go through a complete restart.

Not following these rules can lead to resources remaining assigned and the programmable controller is subsequently in an undefined state.

## 18.3 Common Parameters of the Communication SFCs

### Input Parameter REQ

The input parameter REQ (request to activate) is a level-triggered control parameter. It is used to trigger the job (the data transfer or the connection abort):

- If you call the SFC for a job that is not currently active, you trigger the job with REQ=1. If there is no connection to the communication partner when the communication SFC is called the first time, the connection is established before data transfer begins.
- If you trigger a job and it is not yet completed when you call the SFC again for the same job, REQ is not evaluated by the SFC.

### Output Parameters RET\_VAL and BUSY

The communication SFCs are executed asynchronously, this means that the execution of a job extends over more than one SFC call. The output parameters RET\_VAL and BUSY indicate the status of the job, see Section 2.2.

### Input Parameter CONT

The input parameter CONT (continue) is a control parameter. Using this parameter, you decide whether or not a connection to the communication partner remains established after the job is completed.

- If you select CONT=0 at the first call, the connection is terminated again after the data transfer is completed. The connection is then available again for data exchange with a new communication partner.

This method ensures that connection resources are only occupied when they are actually in use.

- If you select CONT=1 at the first call, the connection remains established on completion of the data transfer.

This method is, for example, useful when you exchange data cyclically between two stations.

---

#### Note

A connection established with CONT=1 can be terminated explicitly with SFC69 "X\_ABORT" or with SFC74 "I\_ABORT".

---

## 18.4 Data Consistency with GET and PUT SFCs

### Access to the Work Memory of the CPU

The communication functions of the operating system access the work memory of the CPU in fields of fixed sizes. The field size depends on the specific CPU and is listed in the following table for the individual CPUs.

Table 18-5 Field Size for Access to the Work Memory of the CPU in Bytes

S7-300 CPUs	CPU 412-1	CPU 413-1/2	CPU 414-1/2	CPU 416-1/2
8	16	16	32	32

This guarantees that the interrupt reaction time is not extended by using communication functions. Since this access is asynchronous to the user program, you cannot transfer any number of bytes of data and be sure of maintaining the consistency of the data.

The rules for ensuring consistency are explained below.

**Consistency Rules for the GET SFCs**

With SFCs 67 “X\_GET” and 72 “I\_GET”, data are transferred consistently if you keep to all the following consistency rules:

- **Active CPU (data receiver):**  
Read out the receive area in the OB in which you call the corresponding SFC.  
If you cannot keep to this rule, then read out the receive area when the execution of the relevant SFC is completed.
- **Passive CPU (data sender):**  
Do not write more data to the send area than the field size of the passive CPU (data sender).
- **Passive CPU (data sender):**  
Write the data to be sent to the send area while interrupts are disabled.

The following diagram shows an example of data transfer for which there can be no guarantee of data consistency because the second consistency rule was not kept to: 16 bytes are transferred although the field size of the passive CPU (data sender) is only 8 bytes.

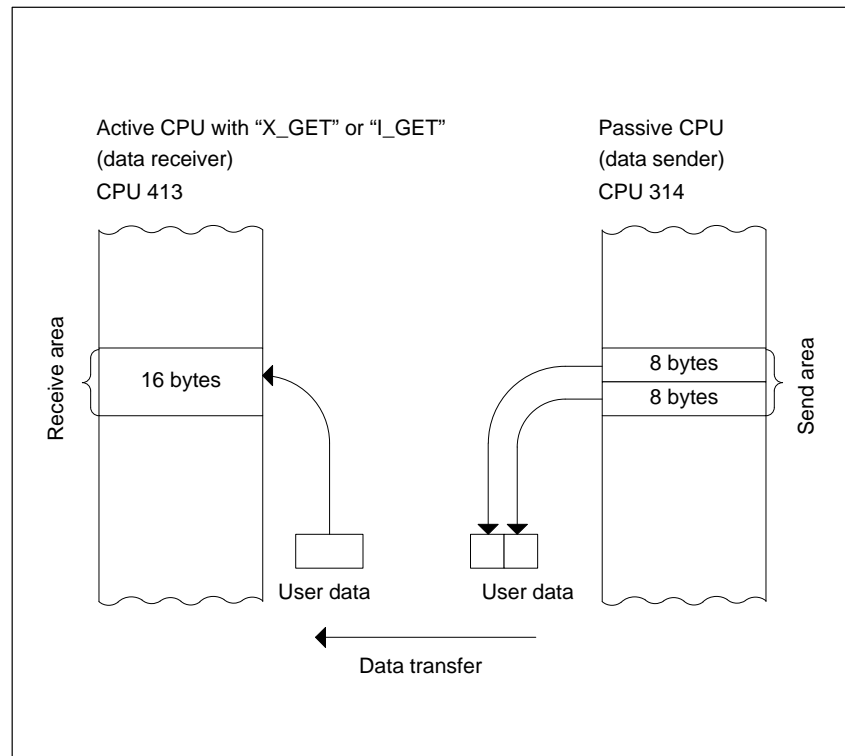


Figure 18-1 Example of a Data Transfer with SFC67 “X\_GET” or SFC72 “I\_GET”, for which no Data Consistency Can Be Guaranteed

**Consistency Rules for the PUT SFCs**

With SFCs 68 “X\_PUT” and 73 “I\_PUT”, data are transferred consistently if you keep to all the following consistency rules:

- **Active CPU (data sender):**  
Write the send area from within the OB in which you call the corresponding SFC.  
If you cannot keep to this rule, then write to the send area only after the first call of the relevant SFC is completed.
- **Active CPU (data sender):**  
Do not write more data to the send area than the field size of the passive CPU (data receiver).
- **Passive CPU (data receiver):**  
Read the received data from the receive area while interrupts are disabled.

The following diagram shows an example of data transfer for which there can be no guarantee of data consistency because the second consistency rule was not kept to: 32 bytes are transferred although the field size of the passive CPU (data receiver) is only 16 bytes.

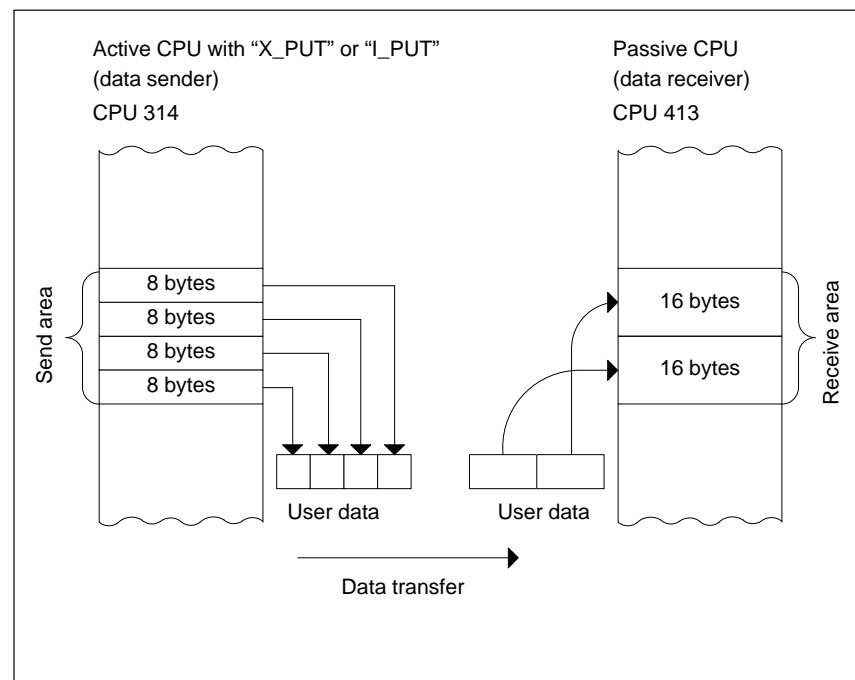


Figure 18-2 Example of a Data Transfer with SFC68 “X\_PUT” or SFC73 “I\_PUT”, for which no Data Consistency Can Be Guaranteed



## 18.5 Sending Data to a Communication Partner outside the Local S7 Station with SFC65 "X\_SEND"

### Description

With SFC65 "X\_SEND", you send data to a communication partner outside the local S7 station. The data is received on the communication partner using SFC66 "X\_RCV".

You can identify your data with the input parameter REQ\_ID. This job ID is sent with the data. You can evaluate this parameter on the communication partner to find out the origin of the data.

The data is sent after calling the SFC with REQ=1.

Make sure that the send area defined by the parameter SD (on the sending CPU) is smaller than or the same size as the receive area defined by the parameter RD (on the communication partner). If SD is of the BOOL data type, RD must also be BOOL.

### Parameters

Table 18-6 Parameters for SFC65 "X\_SEND"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", see 18.3
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "continue", see 18.3
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". This contains the MPI address of the communication partner. You configured this with STEP 7.
REQ_ID	INPUT	DWORD	I, Q, M, D, L, constant	Job identifier. This is used to identify the data on the communication partner.
SD	INPUT	ANY	I, Q, M, D	Reference to the send area. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Sending is not yet completed. BUSY=0: Sending is completed or no send function active.

**Input Parameter  
REQ\_ID**

The input parameter REQ\_ID is used to identify your data. It is passed on by the operating system of the sending CPU to the SFC66 “X\_RCV” of the CPU of the communication partner.

You require the parameter REQ\_ID at the receiving end in the following situations:

- When you call more than one SFC65 “X\_SEND” blocks with different REQ\_ID parameters on the sending CPU and transfer the data to one communication partner.
- When you send data to one communication partner from more than one sending CPU using SFC65 “X\_SEND”.

By evaluating REQ\_ID, you can save the received data in different memory areas.

**Data Consistency**

The data to be sent are copied to an internal buffer of the operating system when the SFC is first called. You must make sure that the send area is not written to before the first call is completed. Otherwise, inconsistent data could be transferred.

Writing to the send area after the first call does not affect the consistency of the data.

**Error Information**

See Table 18-14.

## 18.6 Receiving Data from a Communication Partner outside the Local S7 Station with SFC66 "X\_RCV"

### Description

With SFC66 "X\_RCV", you receive the data sent by one or more communication partners outside the local S7 station using SFC65 "X\_SEND".

With SFC66 "X\_RCV"

- You can check whether data have been sent and are waiting to be copied. The data were entered in an internal queue by the operating system.
- You can copy the oldest block of data from the queue to a selected receive area.

### Parameters

Table 18-7 Parameters for SFC66 "X\_RCV"

Parameter	Declaration	Data Type	Memory Area	Description
EN_DT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "enable data transfer". With the value 0, you can check whether at least one block of data is waiting to be entered in the receive area. The value 1 copies the oldest block of data in the queue to the area of the work memory specified in RD.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains the following: <ul style="list-style-type: none"> <li>• W#16#7000 if EN_DT=0/1 and NDA=0. In this case, there is no data block in the queue.</li> <li>• if EN_DT=0 and NDA=1 the length of the oldest block of data entered in the queue as a positive number in bytes.</li> <li>• if EN_DT=1 and NDA=1 the length of the block of data copied to the RD receive area as a positive number in bytes.</li> </ul>
REQ_ID	OUTPUT	DWORD	I, Q, M, D, L	Job identifier of the SFC "X_SEND" whose data are first in the queue, in other words the oldest data in the queue. If there is no block of data in the queue, REQ_ID has the value 0.

Table 18-7 Parameters for SFC66 “X\_RCV”, continued

Parameter	Declaration	Data Type	Memory Area	Description
NDA	OUTPUT	BOOL	I, Q, M, D, L	Status parameter “new data arrived”. NDA=0: There is no block of data in the queue. NDA=1: • The queue contains at least one block of data. (SFC66 call with EN_DT=0). • The oldest block of data in the queue was copied to the user program. (SFC66 call with EN_DT=1).
RD	OUTPUT	ANY	I, Q, M, D	Reference to the received data area. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL. If you want to discard the oldest block of data in the queue, assign the value NIL to RD.

### Indicating Reception of Data with EN\_DT=0

As soon as data from a communication partner arrive, they are entered in the queue by the operating system in the order in which they are received.

If you want to check whether at least one block of data is in the queue, call SFC66 with EN\_DT=0 and evaluate the output parameter NDA as follows:

- NDA=0 means that the queue does not contain a block of data. REQ\_ID is irrelevant, RET\_VAL has the value W#16#7000.
- NDA=1 means that there is at least one block of data in the queue that can be fetched.

In this case, you should also evaluate the output parameter RET\_VAL and, if applicable, REQ\_ID. RET\_VAL contains the length of the block of data in bytes, REQ\_ID contains the job identifier of the sending block. If there are several blocks of data in the queue, REQ\_ID and RET\_VAL belong to the oldest block of data in the queue.

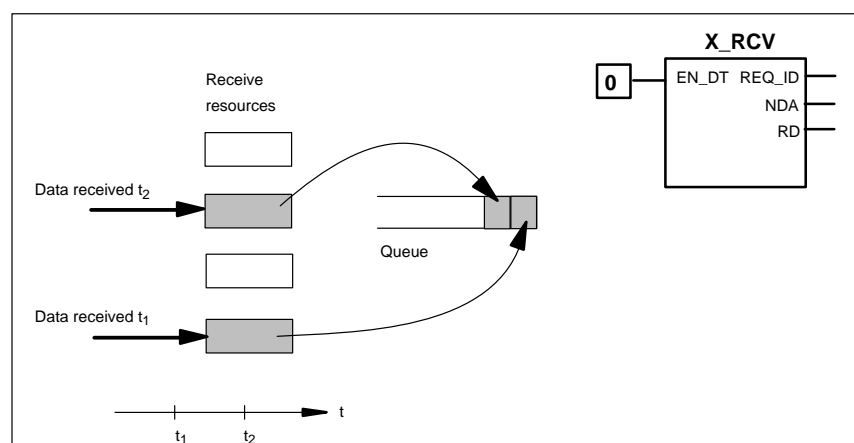


Figure 18-3 Querying Data Reception

### Entering Data in the Receive Area with EN\_DT=1

When you call SFC66 “X\_RCV” with EN\_DT=1, the oldest block of data in the queue is copied to the area of the work memory specified by RD. RD must be larger or the same size as the send area of the corresponding SFC65 “X\_SEND” defined by the SD parameter. If the input parameter SD is of the BOOL data type, RD must also be the BOOL data type.

If you want to enter the received data in different areas, you can query REQ\_ID (SFC call with EN\_DT = 0) and select a suitable RD in the follow-on call (with EN\_DT = 1).

If no error occurs when the data are copied, RET\_VAL contains the length of the copied block of data in bytes and a positive acknowledgment is sent to the sender.

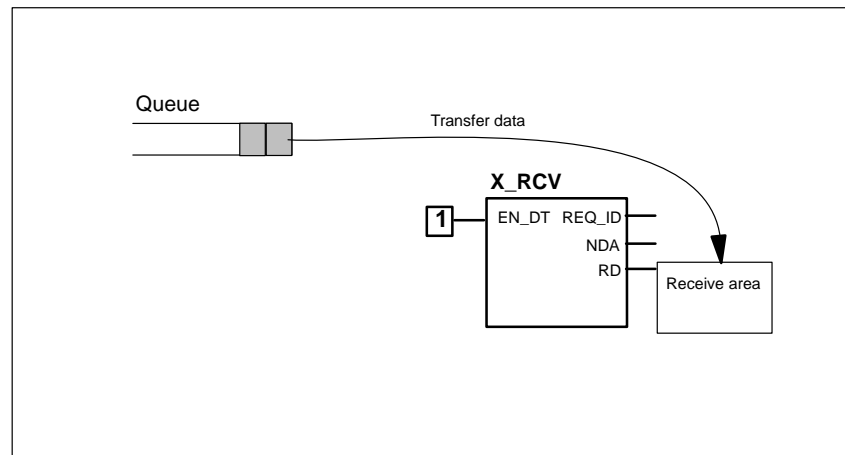


Figure 18-4 Entering Data from the Queue

### Discarding Data

If you do not want to enter the data from the queue, assign the value NIL to RD (see /232/). In this case, the sender receives a negative acknowledgment (RET\_VAL of the corresponding SFC65 “X\_SEND” has the value W#1680B8). RET\_VAL of the SFC66 “X\_RCV” has the value 0.

### Data Consistency

You must not read out the receive area until the job has been completed. Otherwise, you could read data that do not belong together (inconsistent data).

**Changing to the STOP Mode**

If the CPU changes to the STOP mode

- all newly arriving jobs are acknowledged negatively.
- all jobs that have arrived and are in the queue are acknowledged negatively.
  - If the STOP is followed by a complete restart, the blocks of data are all discarded.
  - If the STOP is followed by a restart, (only possible on an S7-400) the block of data belonging to the oldest job is entered in the user program, if the queue was queried before the change to the STOP mode (by calling SFC66 “X\_RCV” with EN\_DT=0). Otherwise it is discarded.

All other blocks of data are discarded.

**Connection Abort**

if the connection is terminated a job belonging to the connection that is already in the queue is discarded.

Exception: If this job is the oldest in the queue, and you have already detected its presence by calling SFC66 “X\_RCV” with EN\_DT=0, you can enter it in the receive area with EN\_DT=1.

**Error Information**

See Table 18-14.

## 18.7 Reading Data from a Communication Partner outside the Local S7 Station with SFC67 "X\_GET"

### Description

With SFC67 "X\_GET", you can read data from a communication partner that is not in the local S7 station. There is no corresponding SFC on the communication partner.

The read job is activated after calling the SFC with REQ=1. Following this, you continue to call the SFC until the data reception is indicated by BUSY=0. RET\_VAL then contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the area to be read as defined by the VAR\_ADDR parameter (on the communication partner). The data types of RD and VAR\_ADDR must also match.

### Parameters

Table 18-8 Parameters for SFC67 "X\_GET"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", see 18.3
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "continue", see 18.3
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". This contains the MPI address of the communication partner. You configured this with STEP 7.
VAR_ADDR	INPUT	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. You must choose a data type that is supported by the communication partner.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.  If no error occurs, RET_VAL contains the length of the block of data copied to the receive area RD as a positive number of bytes.

Table 18-8 Parameters for SFC67 "X\_GET", continued

Parameter	Declaration	Data Type	Memory Area	Description
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Receiving is not yet completed. BUSY=0: Receiving is completed or there is no receive job active.
RD	OUTPUT	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL.  The receive area RD must be at least as long as the area from which the data are read defined by the VAR_ADDR parameter on the communication partner. The data types of RD and VAR_ADDR must also match.

**Data Consistency** See Section 18.4.

#### Changing to the STOP Mode

If the CPU changes to the STOP mode, the connection established by SFC67 "X\_GET" is terminated. Whether or not the received data located in a buffer of the operating system are lost depends on the type of restart following the stoppage:

- Following a restart (only on the S7-400) the data are copied to the area defined by RD.
- Following a complete restart, the data are discarded.

#### Partner Changes to the STOP Mode

If the CPU of the communication partner changes to the STOP mode, this does not affect the data transfer with SFC67 "X\_GET". The data can also be read with the partner in the STOP mode.

**Error Information** See Table 18-14.



## 18.8 Writing Data to a Communication Partner outside the Local S7 Station with SFC68 “X\_PUT”

### Description

With SFC68 “X\_PUT”, you write data to a communication partner that is not in the same local S7 station. There is no corresponding SFC on the communication partner.

The write job is activated after calling the SFC with REQ=1. Following this, you continue to call the SFC until the acknowledgment is received with BUSY=0.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR\_ADDR parameter (on the communication partner). The data types of SD and VAR\_ADDR must also match.

### Parameters

Table 18-9 Parameters for SFC68 “X\_PUT”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter “request to activate”, see 18.3
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter “continue”, see 18.3
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter “destination ID”. This contains the MPI address of the communication partner. You configured this with STEP 7.
VAR_ADDR	INPUT	ANY	I, Q, M, D	Reference to the area on the partner CPU to which the data will be written. You must choose a data type that is supported by the communication partner.
SD	INPUT	ANY	I, Q, M, D	Reference to the area in the local CPU that contains the data to be sent. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL. SD must have the same length as the VAR_ADDR parameter of the communication partner. The data types of SD and VAR_ADDR must also match.

Table 18-9 Parameters for SFC68 "X\_PUT", continued

Parameter	Declaration	Data Type	Memory Area	Description
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Sending is not yet completed. BUSY=0: Sending is completed or no send function active.

**Data Consistency** See Section 18.4.

**Changing to the STOP Mode** If the CPU changes to the STOP mode, the connection established by SFC68 "X\_PUT" is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the content of the buffer is discarded.

**Partner Changes to the STOP Mode** If the CPU of the communication partner changes to the STOP mode, this does not affect the data transfer with SFC68 "X\_PUT". The data can also be written with the partner in the STOP mode.

**Error Information** See Table 18-14.

## 18.9 Aborting an Existing Connection to a Communication Partner outside the Local S7 Station with SFC 69 "X\_ABORT"

### Description

With SFC69 "X\_ABORT", you terminate a connection that was established by SFCs X\_SEND, X\_GET or X\_PUT to a communication partner that is not in the same local S7 station. At the same time, the resources used for the connection are released at both ends.

You can only call SFC69 "X\_ABORT" at the end where the SFCs "X\_SEND", "X\_PUT" or "X\_GET" are located.

The connection abort is activated by calling the SFC with REQ=1.

### Parameters

Table 18-10 Parameters for SFC69 "X\_ABORT"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", see 18.3
DEST_ID	INPUT	WORD	I, Q, M, D, L, constant	Address parameter "destination ID". This contains the MPI address of the communication partner. You configured this with STEP 7.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: The connection abort is not yet completed. BUSY=0: the connection abort is completed.

### Changing to the STOP Mode

If the CPU changes to the STOP mode, the connection abort started with SFC69 "X\_ABORT" is completed.

### Partner Changes to the STOP Mode

If the CPU of the communication partner changes to the STOP mode, this does not affect the connection abort with SFC69 "X\_ABORT". The connection is terminated.

### Error Information

See Table 18-14.

## 18.10 Reading Data from a Communication Partner within the Local S7 Station with SFC72 "I\_GET"

### Description

With SFC72 "I\_GET", you can read data from a communication partner in the same local S7 station. The communication partner can be in the central rack, in an expansion rack or distributed. Make sure that you assign distributed communication partners to the local CPU with STEP 7. There is no corresponding SFC on the communication partner.

The receive job is activated after calling the SFC with REQ=1. Following this, you continue to call the SFC until the data reception is indicated by BUSY=0. RET\_VAL then contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the area to be read as defined by the VAR\_ADDR parameter (on the communication partner). The data types of RD and VAR\_ADDR must also match.

### Parameters

Table 18-11 Parameters for SFC72 "I\_GET"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", see 18.3
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "continue", see 18.3
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier of the address range on the partner module: B#16#54= Peripheral input (PI) B#16#55= Peripheral output (PQ) The identifier of a range belonging to a mixed module is the lower of the two addresses. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the partner module. If it is a mixed module, specify the lower of the two addresses.
VAR_ADDR	INPUT	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. Select a data type supported by the communication partner.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.  If no error occurs, RET_VAL contains the length of the block of data copied to the receive area RD as a positive number of bytes.

Table 18-11 Parameters for SFC72 “I\_GET”, continued

Parameter	Declaration	Data Type	Memory Area	Description
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Receiving is not yet completed. BUSY=0: Receiving is completed or there is no receive job active.
RD	OUTPUT	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL. The receive area RD must be at least as long as the area from which the data are read defined by the VAR_ADDR parameter on the communication partner. The data types of RD and VAR_ADDR must also match.

**Data Consistency**      See Section 18.4.

**Changing to the STOP Mode**      If the CPU changes to the STOP mode, the connection established by SFC72 “I\_GET” is terminated. Whether or not the received data located in a buffer of the operating system are lost depends on the type of restart following the stoppage:

- Following a restart (only on the S7-400) the data are copied to the area defined by RD.
- Following a complete restart, the data are discarded.

**Partner Changes to the STOP Mode**      If the CPU of the communication partner changes to the STOP mode, this does not affect the data transfer with SFC72 “I\_GET”. The data can also be read with the partner in the STOP mode.

**Error Information**      See Table 18-14.

## 18.11 Writing Data to a Communication Partner within the Local S7 Station with SFC73 “I\_PUT”

### Description

With SFC73 “I\_PUT”, you write data to a communication partner that is in the same local S7 station.. The communication partner can be in the central rack, in an expansion rack or distributed. Make sure that you assign distributed communication partners to the local CPU with STEP 7. There is no corresponding SFC on the communication partner.

The send job is activated after calling the SFC with signal level 1 at the REQ control input.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR\_ADDR parameter (on the communication partner). The data types of SD and VAR\_ADDR must also match.

### Parameters

Table 18-12 Parameters for SFC73 “I\_PUT”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter “request to activate”, see 18.3
CONT	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter “continue”, see 18.3
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier of the address range on the partner module: B#16#54= Peripheral input (PI) B#16#55= Peripheral output (PQ) The identifier of a range belonging to a mixed module is the lower of the two addresses. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the partner module. If it is a mixed module, specify the lower of the two addresses.
VAR_ADDR	INPUT	ANY	I, Q, M, D, L	Reference to the area on the communication partner to which the data will be written. Choose a data type that is supported by the communication partner.

Table 18-12 Parameters for SFC73 “I\_PUT”, continued

Parameter	Declaration	Data Type	Memory Area	Description
SD	INPUT	ANY	I, Q, M, D	Reference to the area on the local CPU that contains the data to be sent. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of these data types except for BOOL. SD must be the same length as the parameter VAR_ADDR of the communication partner. The data types of SD and VAR_ADDR must also match.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: Sending is not yet completed. BUSY=0: Sending is completed or no send function active.

**Data Consistency**      See Section 18.4.

**Changing to the STOP Mode**      If the CPU changes to the STOP mode, the connection established by SFC73 “I\_PUT” is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the content of the buffer is discarded.

**Partner Changes to the STOP Mode**      If the CPU of the communication partner changes to the STOP mode, this does not affect the data transfer with SFC73 “I\_PUT”. The data can also be written with the partner in the STOP mode.

**Error Information**      See Table 18-14.

## 18.12 Aborting an Existing Connection to a Communication Partner within the Local S7 Station with SFC 74 "I\_ABORT"

### Description

With SFC74 "I\_ABORT", you terminate a connection that was established by SFC72 "I\_GET" or SFC73 "I\_PUT" to a communication partner in the same local S7 station,. At the same time, the resources used for the connection are released at both ends.

You can only call SFC74 "I\_ABORT" at the end where the SFC "I\_PUT" or "I\_GET" is located (in other words at the client end).

The connection abort is activated by calling the SFC with REQ=1.

### Parameters

Table 18-13 Parameters for SFC74 "I\_ABORT"

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M, D, L, constant	Control parameter "request to activate", see 18.3
IOID	INPUT	BYTE	I, Q, M, D, L, constant	Identifier of the address range on the partner module: B#16#54= Peripheral input (PI) B#16#55= Peripheral output (PQ) The identifier of a range belonging to a mixed module is the lower of the two addresses. If the addresses are the same, specify B#16#54.
LADDR	INPUT	WORD	I, Q, M, D, L, constant	Logical address of the partner module. If it is a mixed module, specify the lower of the two addresses.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	If an error occurs while the function is being executed, the return value contains the corresponding error code.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: The connection abort is not yet completed. BUSY=0: the connection abort is completed.

### Changing to the STOP Mode

If the CPU changes to the STOP mode, the connection abort started with SFC74 "I\_ABORT" is completed.

### Partner Changes to the STOP Mode

If the CPU of the communication partner changes to the STOP mode, this does not affect the connection abort with SFC74 "I\_ABORT". The connection is terminated.

### Error Information

See Table 18-14.



### 18.13 Error Information of the Communication SFCs for Non-Configured Connections

**Error Information** The “real” error information for SFCs 65 to 74 as shown in Table 18-14 can be classified as follows:

Error Code (W#16# ...)	Explanation
809x	Error on the CPU on which the SFC is executed
80Ax	Permanent communication error
80Bx	Error on the communication partner
80Cx	Temporary error

Table 18-14 Specific Error Information for SFCs 65 to 74

Error Code (W#16# ...)	Explanation (general)	Explanation (for specific SFC)
0000	Execution completed without errors.	SFC69 “X_ABORT” and SFC74 “I_ABORT”:REQ=1, and the specified connection is not established. SFC66 “X_RCV”: EN_DT=1 and RD=NIL
00xy	—	SFC66 “X_RCV” with NDA=1 and RD<>NIL:RET_VAL contains the length of the received data (with EN_DT=0) or the length of the data copied to RD (with EN_DT=1). SFC67 “X_GET”:RET_VAL contains the length of the received block of data. SFC72 “I_GET”:RET_VAL contains the length of the received block of data.
7000	—	SFC65 “X_SEND”, SFC67 “X_GET”, SFC68 “X_PUT”, SFC69 “X_ABORT”, SFC72 “I_GET”, SFC73 “I_PUT” and SFC74 “I_ABORT”: call with REQ = 0 (call without execution), BUSY has the value 0, no data transfer active. SFC66 “X_RCV”: EN_DT=0/1 and NDA=0
7001	First call with REQ=1: data transfer was triggered; BUSY has the value 1.	—
7002	Interim call (REQ irrelevant): data transfer is already active ; BUSY has the value 1.	—

Table 18-14 Specific Error Information for SFCs 65 to 74, continued

Error Code (W#16# ...)	Explanation (general)	Explanation (for specific SFC)
8090	Specified destination address of the communication partner is invalid, for example: <ul style="list-style-type: none"> <li>wrong IOID</li> <li>wrong base address exists</li> <li>wrong MPI address (&gt; 126)</li> </ul>	—
8092	Error in SD or RD, for example: addressing the local data area is not permitted.	SFC65 “X_SEND”, for example <ul style="list-style-type: none"> <li>illegal length for SD</li> <li>SD=NIL is illegal</li> </ul>
		SFC66 “X_RCV”, for example <ul style="list-style-type: none"> <li>More data were received than can fit in the area specified by RD.</li> <li>RD is of the BOOL data type, the received data is, however, longer than a byte.</li> </ul>
		SFC67 “X_GET” and SFC72 “I_GET”, for example <ul style="list-style-type: none"> <li>illegal length for RD</li> <li>the length or the data type of RD does not match the received data.</li> <li>RD=NIL is not permitted.</li> </ul>
		SFC68 “X_PUT” and SFC73 “I_PUT”, for example <ul style="list-style-type: none"> <li>illegal length for SD</li> <li>SD=NIL is illegal</li> </ul>
8095	The block is already being executed in a lower priority class.	—
80A0	Error in the received acknowledgment	SFC68 “X_PUT” and SFC73 “I_PUT”: The data type specified in the SD of the sending CPU is not supported by the communication partner.
80A1	Communication problems: SFC call after terminating an existing connection	—
80B0	Object is not obtainable, for example DB not loaded	Possible with SFC67 “X_GET” and SFC68 “X_PUT” and SFC72 “I_GET” and SFC73 “I_PUT”
80B1	Error in the ANY pointer. The length of the data area to be sent is incorrect.	—
80B2	Hardware error: module does not exist <ul style="list-style-type: none"> <li>The configured slot is not occupied.</li> <li>Actual module type does not match expected type</li> <li>Distributed peripheral I/Os not available.</li> <li>No entry for the module in the corresponding SDB.</li> </ul>	Possible with SFC67 “X_GET” and SFC68 “X_PUT” and SFC72 “I_GET” and SFC73 “I_PUT”
80B3	Data can either only be read or only written, for example write-protected DB	Possible with SFC67 “X_GET” and SFC68 “X_PUT” and SFC72 “I_GET” and SFC73 “I_PUT”

Table 18-14 Specific Error Information for SFCs 65 to 74, continued

Error Code (W#16# ...)	Explanation (general)	Explanation (for specific SFC)
80B4	Data type error in the ANY pointer, or ARRAY of the specified type not allowed.	SFC67 "X_GET" and SFC68 "X_PUT" and SFC72 "I_GET" and SFC73 "I_PUT": The data type specified in VAR_ADDR is not supported by the communication partner.
80B5	Execution rejected due to illegal mode	Possible with SFC65 "X_SEND"
80B6	The received acknowledgment contains an unknown error code.	—
80B7	Data type and/or length of the transferred data does not fit in the area on the partner CPU to which it should be written.	Possible with SFC68 "X_PUT" and SFC73 "I_PUT"
80B8	—	SFC65 "X_SEND": The SFC66 "X_RCV" of the communication partner did not allow data acceptance (RD=NIL).
80B9	—	SFC65 "X_SEND": The block of data was identified by the communication partner (SFC66 "X_RCV" call with EN_DT=0), it has not yet been entered in the user program because the partner is in the STOP mode.
80BA	The response of the communication partner does not fit in the communication frame.	—
80C0	The specified connection is being used by another job.	—
80C1	Lack of resources on the CPU on which the SFC is executed, for example: <ul style="list-style-type: none"> <li>The maximum number of different send jobs is already being executed on the module.</li> <li>The connection resource is in use, for example to receive data.</li> </ul>	—
80C2	Temporary lack of resources on the communication partner, for example: <ul style="list-style-type: none"> <li>The communication partner is currently processing the maximum number of jobs.</li> <li>The required resources, memory etc. are being used.</li> <li>Not enough work memory. (compress memory).</li> </ul>	—
80C3	Error in connection establishment, for example: <ul style="list-style-type: none"> <li>The local S7 station is not attached to the MPI subnet.</li> <li>You have addressed your own station on the MPI subnet.</li> <li>The communication partner is no longer obtainable.</li> <li>Temporary lack of resources on the communication partner</li> </ul>	—



# Generating Block-Related Messages

# 19

## Chapter Overview

Section	Description	Page
19.1	Introduction to Generating Block-Related Messages with SFBs	19-2
19.2	Generating Block-Related Messages without Acknowledgment with SFB36 “NOTIFY”	19-4
19.3	Generating Block-Related Messages with Acknowledgment with SFB33 “ALARM”	19-6
19.4	Generating Block-Related Messages with Accompanying Values for Eight Signals with SFB35 “ALARM_8P”	19-9
19.5	Generating Block-Related Messages without Accompanying Values for Eight Signals with SFB34 “ALARM_8”	19-11
19.6	Sending Archive Data with SFB37 “AR_SEND”	19-13
19.7	Disabling Block-Related, Symbol-Related and Group Status Messages with SFC10 “DIS_MSG”	19-15
19.8	Enabling Block-Related, Symbol-Related and Group Status Messages with SFC9 “EN_MSG”	19-17
19.9	Startup Behavior of the SFBs for Generating Block-Related Messages	19-19
19.10	Error Response of the SFBs for Generating Block-Related Messages	19-20
19.11	Introduction to Generating Block-Related Messages with SFCs	19-21
19.12	Generating Acknowledgable Block-Related Messages with SFC17 “ALARM_SQ” and Permanently Acknowledged Block-Related Messages with SFC18 “ALARM_S”	19-23
19.13	Querying the Acknowledgment Status of the Last ALARM_SQ Entering State Message with SFC19 “ALARM_SC”	19-26

## 19.1 Introduction to Generating Block-Related Messages With SFBs

### SFBs for Generating Block-Related Messages

You can generate a block-related message by calling one of the following SFBs in your program:

- SFB36 “NOTIFY”
- SFB33 “ALARM”
- SFB35 “ALARM\_8P”
- SFB34 “ALARM\_8”

These SFBs have the following properties:

- Like SFBs for configured connections, they have internal states. The user can use SFC62 “CONTROL” on the SFBs to create block-related messages.
- Each detected edge change causes a message to be sent.
- The accompanying values (inputs SD\_i) are read consistently when the edge is evaluated and assigned to the message.

---

#### Note

The parameters ID, EV\_ID, SEVERITY, and SD\_i are only evaluated the first time the block is called (the actual parameters or the predefined values from the instance).

---

### Logging On Display Devices

Before SFBs for generating block-related messages can send a message when an edge change is detected, at least one display device must be logged on for block-related messages.

### Storing Messages

To avoid messages being lost even when there is a lot of traffic on the communication system, each SFB that generates a message can buffer two messages.

If, however, messages are lost, you are informed of this by the ERROR and STATUS output parameters (ERROR = 0, STATUS = 11). The next time that a message can be sent, the logged on display devices are also informed of the loss.

**Acknowledging Messages**

A centralized acknowledgment concept is used. When you have acknowledged the message at a display device, the acknowledgment information is first sent to the CPU that generated the message. From here, the acknowledgment information is distributed to all stations logged on for this purpose. You acknowledge a signal and not an individual message. If, for example, several rising edges of a signal were indicated and you acknowledge the event entering the state, all previous events with the same message number count as having been acknowledged.

**Acknowledgment Display**

SFB36 "NOTIFY" does not have an acknowledgment display. You can check the output parameters ACK\_UP and ACK\_DN of SFB33 "ALARM" and the output parameter ACK\_STATE of SFBs 35 "ALARM\_8P" and "ALARM\_8". These outputs are updated when the block is called providing the control parameter EN\_R has the value 1.

**Disabling and Enabling Messages**

In some situations, it may be useful to suppress messages, for example when a signal is "fluttering" or when you start up your system. You can therefore disable and enable messages at the display device or in your program. Disabling/enabling applies to all stations that logged on for the particular message. A disabled message remains disabled until it is enabled again. You are informed of disabled messages with the ERROR and STATUS output parameters (ERROR = 1, STATUS = 21).

**Message Update**

With a message update, you can read out the current signal and acknowledgment states at a display device. During the update, all the logged on stations continue to receive the messages for which they logged on.

**Amount of Transferable data**

The data transferred with the accompanying values SD\_i of the NOTIFY, ALARM and ALARM\_8P SFBs must not exceed a maximum length. The maximum data length is calculated as follows:

$$\text{maxleng} = \min(\text{pdu\_local}, \text{pdu\_remote}) - 44 - 4 * \text{number of SD\_i parameters used}$$

Where:

- pdu\_local is the maximum length of the data fields of the local CPU (SZL\_ID W#16#0131, INDEX 1, variable pdu)
- pdu\_remote is the maximum length of data fields of the display devices

Example: A CPU 414-1 is sending messages to a PG 760 (via MPI). The accompanying values SD\_1, SD\_2 and SD\_3 are used.

pdu\_local = 480 bytes, pdu\_remote = 480 bytes,  
number of SD\_i parameters used: 3

So that:

$$\text{maxleng} = \min(480, 480) - 44 - 4 * 3 = 480 - 44 - 12 = 424$$

The maximum length of data that can be transferred per SFB is 424 bytes.

## 19.2 Generating Block-Related Messages Without Acknowledgment with SFB36 “NOTIFY”

### Description

SFB36 “NOTIFY” monitors a signal. It generates a message both on a rising edge (event entering state) and on a falling edge (event leaving state). You can have up to ten accompanying values sent with the message. The message is sent to all stations logged on for this purpose. When the SFB is first called, a message with the current signal state is sent.

The accompanying values are queried when the edge is detected and assigned to the message. If you acknowledge such a message at a logged on display device, all other logged on display devices are informed. The NOTIFY block is not informed of this acknowledgment.

SFB36 “NOTIFY” can temporarily store one rising and one falling signal edge. Any further signal changes that occur are ignored. This loss of messages is indicated with the output parameters ERROR and STATUS (ERROR = 0, STATUS = 11); the logged on display devices are also informed of this loss.

SFB36 “NOTIFY” complies with the IEC 1131-5 standard.

### Parameters

Table 19-1 Parameters for SFB36 “NOTIFY”

Parameter	Declaration	Data Type	Memory Area	Description
SIG	INPUT	BOOL	I, Q, M, D, L, constant	The signal to be monitored
ID	INPUT	WORD	I, Q, M, D, L, constant	Data channel for messages: W#16#EEEE ID is only evaluated at the first call.
EV_ID	INPUT	DWORD	I, Q, M, D, L, constant	Message number (0 not permitted) EV_ID is only evaluated at the first call. Following this, each time SFB36 is called with the corresponding instance DB, the message number of the first call is used. When assigning the message number, use the message configuration functions. This ensures the consistency of the message numbers.
SEVERITY	INPUT	INT	I, Q, M, D, L, constant	Weighting of the event Possible values: 0 through 127
DONE	OUTPUT	BOOL	I, Q, M, D, L	DONE status parameter: Generation of message completed
ERROR	OUTPUT	BOOL	I, Q, M, D, L	ERROR status parameter
STATUS	OUTPUT	INT	I, Q, M, D, L	STATUS status parameter
SD <sub>i</sub> , 1 ≤ i ≤ 10	IN_OUT	ANY	I, Q, M, D, T, C	Accompanying value The following data types are permitted BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.



**Error Information**

Table 19-2 contains all the error information specific to SFB36 that can be output with the ERROR and STATUS parameters.

Table 19-2 Error Information for SFB36 "NOTIFY"

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job not active because the previous job is not yet completed.
0	22	Warning: error in pointer to accompanying values (SD_i) involving the data length or data type. The activated message will be sent.
1	1	Communications problems
1	3	No logon for the specified EV_ID
1	4	<ul style="list-style-type: none"> <li>• The specified EV_ID is outside the permitted range or</li> <li>• The maximum memory area that can be sent for the CPU per SFB36 was exceeded</li> <li>• SEVERITY is not in the permitted range.</li> </ul>
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: <ul style="list-style-type: none"> <li>• an uninitialized instance DB was specified</li> <li>• an instance DB that does not belong to SFB36 was specified</li> <li>• a shared DB instead of an instance DB was specified</li> </ul>
1	20	Not enough memory.
1	21	The message with the specified EV_ID is disabled

### 19.3 Generating Block-Related Messages With Acknowledgment with SFB33 “ALARM”

#### Description

SFB33 “ALARM” monitors a signal. It generates a message both on a rising edge (event entering state) and on a falling edge (event leaving state). You can have up to ten accompanying values sent with the message. The message is sent to all stations logged on for this purpose. When the SFB is first called, a message with the current signal state is sent.

The ACK\_UP output is reset when there is a rising edge and is set when your acknowledgment of the event entering the state has arrived from a logged on display device. The situation for the ACK\_DN output is analogous: this is reset when there is a falling edge and is set when your acknowledgment of the event leaving the state is received from a logged on display device. Once your acknowledgment has been received from a logged on display device, the acknowledgment information is passed on to all other stations logged on for this purpose.

SFB33 “ALARM” can temporarily store one rising and one falling signal edge. Any further signal changes that occur are ignored. This loss of messages is indicated with the output parameters ERROR and STATUS (ERROR = 0, STATUS = 11); the logged on display devices are also informed of this loss.

SFB33 “ALARM” complies with the IEC 1131-5 standard.

## Parameters

Table 19-3 Parameters for SFB33 “ALARM”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M,D, L, constant	Control parameter enabled to receive that decides whether the outputs ACK_UP and ACK_DN are updated at the first block call (EN_R=1) or not (EN_R=0). If EN_R=0 SFB33 “ALARM” operates in the same way as SFB36 “NOTIFY”. The output parameters ACK_UP and ACK_DN remain the same in this case.
SIG	INPUT	BOOL	I, Q, M,D, L, constant	The signal to be monitored
ID	INPUT	WORD	I, Q, M,D, L, constant	Data channel for messages: W#16#EEEE ID is only evaluated at the first call.
EV_ID	INPUT	DWORD	I, Q, M,D, L, constant	Message number (0 not permitted) EV_ID is only evaluated at the first call. Following this, each time SFB33 is called with the corresponding instance DB, the message number of the first call is used. When assigning the message number, use the message configuration functions. This ensures the consistency of the message numbers.
SEVERITY	INPUT	INT	I, Q, M,D, L, constant	Weighting of the event Possible values: 0 through 127
DONE	OUTPUT	BOOL	I, Q, M,D, L	DONE status parameter: Generation of message completed
ERROR	OUTPUT	BOOL	I, Q, M,D, L	ERROR status parameter
STATUS	OUTPUT	INT	I, Q, M,D, L	STATUS status parameter
ACK_DN	OUTPUT	BOOL	I, Q, M,D, L	Event leaving state was acknowledged on a display device
ACK_UP	OUTPUT	BOOL	I, Q, M,D, L	Event entering state was acknowledged on a display device
SD_i, 1 ≤ i ≤ 10	IN_OUT	ANY	I, Q, M,D, T, C	Accompanying value The following data types are permitted BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

**Error Information**

Table 19-4 contains all the error information specific to SFB33 that can be output with the ERROR and STATUS parameters.

Table 19-4 Error Information for SFB33 “ALARM”

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job not active because the previous job is not yet completed.
0	22	Warning: error in pointer to accompanying values (SD_i) involving the data length or data type. The activated message will be sent.
1	1	Communications problems
1	3	No logon for the specified EV_ID
1	4	<ul style="list-style-type: none"><li>• The specified EV_ID is outside the permitted range or</li><li>• The maximum memory area that can be sent for the CPU per SFB33 was exceeded</li><li>• SEVERITY is not in the permitted range.</li></ul>
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: <ul style="list-style-type: none"><li>• an uninitialized instance DB was specified</li><li>• an instance DB that does not belong to SFB33 was specified</li><li>• a shared DB instead of an instance DB was specified</li></ul>
1	20	Not enough memory.
1	21	The message with the specified EV_ID is disabled

---

**Note**

After the first block call, the ACK\_UP and ACK\_DN outputs have the value 1 and it is assumed that the previous value of the SIG input was 0.

---

## 19.4 Generating Block-Related Messages with Accompanying Values for Eight Signals with SFB35 “ALARM\_8P”

### Description

SFB35 “ALARM\_8P” is a straight extension of SFB33 “ALARM” allowing eight signals.

A message is generated when an edge change is detected at one or more signals (exception: a message is always sent at the first block call). All eight signals have a common message number that is broken down into eight individual messages on the display device. You can acknowledge each individual message separately or all eight individual messages at once.

You can use the ACK\_STATE output parameter to incorporate the acknowledgment state of the individual messages in your program.

If you disable or enable a message of an ALARM\_8P block, this always affects the entire ALARM\_8P block. Disabling and enabling of individual signals is not possible.

SFB35 “ALARM\_8P” can temporarily store two messages. Any further signal changes are then ignored. This loss of messages is indicated by the ERROR and STATUS output parameters (ERROR = 0, STATUS = 11); the logged on display devices are also informed of the loss.

### Parameters

Table 19-5 Parameters for SFB35 “ALARM\_8P”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M,D, L, constant	Control parameter enabled to receive that decides whether the output ACK_STATE is updated at the block call (EN_R=1) or not (EN_R=0).
SIG_i, 1 ≤ i ≤ 8	INPUT	BOOL	I, Q, M,D, L, constant	i(th) signal to be monitored
ID	INPUT	WORD	I, Q, M,D, L, constant	Data channel for messages: W#16#EEEE ID is only evaluated at the first call.
EV_ID	INPUT	DWORD	I, Q, M,D, L, constant	Message number (0 not permitted) EV_ID is only evaluated at the first call. Following this, each time SFB35 is called with the corresponding instance DB, the message number of the first call is used. When assigning the message number, use the message configuration functions. This ensures the consistency of the message numbers.
SEVERITY	INPUT	INT	I, Q, M,D, L, constant	Weighting of the event Possible values: 0 through 127
DONE	OUTPUT	BOOL	I, Q, M,D, L	DONE status parameter: Generation of message completed
ERROR	OUTPUT	BOOL	I, Q, M,D, L	ERROR status parameter
STATUS	OUTPUT	INT	I, Q, M,D, L	STATUS status parameter

Table 19-5 Parameters for SFB35 "ALARM\_8P", continued

Parameter	Declaration	Data Type	Memory Area	Description
ACK_STATE	OUTPUT	WORD	I, Q, M,D, L	Bit field with the current acknowledgment status of all eight messages: <ul style="list-style-type: none"> <li>• Bit 2<sup>0</sup>: event entering state at SIG_1 was acknowledged</li> <li>• Bit 2<sup>7</sup>: event entering state at SIG_8 was acknowledged</li> <li>• Bit 2<sup>8</sup>: event leaving state at SIG_1 was acknowledged</li> <li>• Bit 2<sup>15</sup>: event leaving state at SIG_8 was acknowledged</li> </ul>
SD_j, 1 ≤ j ≤ 10	IN_OUT	ANY	I, Q, M,D, T, C	Accompanying value These values apply to all messages. The following data types are permitted BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.

**Error Information**

Table 19-6 contains all the error information specific to SFB35 that can be output with the ERROR and STATUS parameters.

Table 19-6 Error Information for SFB35 "ALARM\_8P"

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job not active because old job not completed.
0	22	Warning: error in pointer to accompanying values (SD_i) involving the data length or data type. The activated message will be sent.
1	1	Communications problems
1	3	No logon for the specified EV_ID
1	4	<ul style="list-style-type: none"> <li>• The specified EV_ID is outside the permitted range or</li> <li>• The maximum memory area that can be sent for the CPU per SFB35 was exceeded</li> <li>• SEVERITY is not in the permitted range.</li> </ul>
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: <ul style="list-style-type: none"> <li>• an uninitialized instance DB was specified</li> <li>• an instance DB that does not belong to SFB35 was specified</li> <li>• a shared DB instead of an instance DB was specified</li> </ul>
1	20	Not enough memory.
1	21	The message with the specified EV_ID is disabled

**Note**

After the first block call, all the bits of the ACK\_STATE output are set and it is assumed that the previous values of inputs SIG\_i, 1 ≤ i ≤ 8 were 0.

## 19.5 Generating Block-Related Messages without Accompanying Values for Eight Signals with SFB34 “ALARM\_8”

**Description** SFB34 “ALARM\_8” is identical to SFB35 “ALARM\_8P” except that it does not have the accompanying values SD\_1 through SD\_10.

### Parameters

Table 19-7 Parameters for SFB34 “ALARM\_8”

Parameter	Declaration	Data Type	Memory Area	Description
EN_R	INPUT	BOOL	I, Q, M,D, L, constant	Control parameter enabled to receive that decides whether the output ACK_STATE is updated (EN_R=1) when the block is called or not (EN_R=0).
SIG_i, $1 \leq i \leq 8$	INPUT	BOOL	I, Q, M,D, L, constant	i(th) signal to be monitored
ID	INPUT	WORD	I, Q, M,D, L, constant	Data channel for messages: W#16#EEEE ID is only evaluated at the first call.
EV_ID	INPUT	DWORD	I, Q, M,D, L, constant	Message number (0 not permitted) EV_ID is only evaluated at the first call. Following this, each time SFB34 is called with the corresponding instance DB, the message number of the first call is used. When assigning the message number, use the message configuration functions. This ensures the consistency of the message numbers.
SEVERITY	INPUT	INT	I, Q, M,D, L, constant	Weighting of the event Possible values: 0 through 127
DONE	OUTPUT	BOOL	I, Q, M,D, L	DONE status parameter: Generation of message completed
ERROR	OUTPUT	BOOL	I, Q, M,D, L	ERROR status parameter
STATUS	OUTPUT	INT	I, Q, M,D, L	STATUS status parameter
ACK_STATE	OUTPUT	WORD	I, Q, M,D, L	Bit field with the current acknowledgment status of all eight messages: <ul style="list-style-type: none"> <li>• Bit 2<sup>0</sup>: event entering state at SIG_1 was acknowledged</li> <li>• Bit 2<sup>7</sup>: event entering state at SIG_8 was acknowledged</li> <li>• Bit 2<sup>8</sup>: event leaving state at SIG_1 was acknowledged</li> <li>• Bit 2<sup>15</sup>: event leaving state at SIG_8 was acknowledged</li> </ul>

**Error Information**

Table 19-8 contains all the error information specific to SFB34 that can be output with the ERROR and STATUS parameters.

Table 19-8 Error Information for SFB34 “ALARM\_8”

ERROR	STATUS (decimal)	Explanation
0	11	Warning: new job not active because the previous job is not yet completed.
1	1	Communications problems
1	3	No logon for the specified EV_ID
1	4	The specified EV_ID is outside the permitted range or SEVERITY is not in the permitted range.
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: <ul style="list-style-type: none"><li>• an uninitialized instance DB was specified</li><li>• an instance DB that does not belong to SFB34 was specified</li><li>• a shared DB instead of an instance DB was specified</li></ul>
1	20	Not enough memory.
1	21	The message with the specified EV_ID is disabled

---

**Note**

After the first block call, all the bits of the ACK\_STATE output are set and it is assumed that the previous values of inputs SIG\_i,  $1 \leq i \leq 8$  were 0.

---



## 19.6 Sending Archive Data with SFB37 “AR\_SEND”

### Description

SFB37 “AR\_SEND” sends archive data to operator interface systems logged on for this purpose. These systems inform the CPU of the relevant archive number in the logon message. Depending on the memory available on the CPU and the address area used, the archive data can be up to 65534 bytes long.

The sending of the data is activated by a positive edge at control input REQ after the block has been called. The start address of the archive data is specified by SD\_1, the length of the data field by LEN. Data transfer is asynchronous to the execution of the user program. Successful completion of the transfer is indicated by the DONE status parameter having the value 1. A rising edge at control input R aborts the transfer of data.

### Parameter

Table 19-9 Parameters for SFB37 “AR\_SEND”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M,D, L, constant	Control parameter request
R	INPUT	BOOL	I, Q, M,D, L, constant	Control parameter reset: current job aborted
ID	INPUT	WORD	I, Q, M,D, L, constant	Data channel for messages: W#16#EEEE ID is only evaluated at the first call.
AR_ID	INPUT	DWORD	I, Q, M,D, L, constant	Archive number (0 not permitted) AR_ID is only evaluated at the first call. Following this, each time SFB37 is called with the corresponding instance DB, the archive number from the first call is used. When assigning the archive number, use the message configuration functions. This ensures the consistency of the archive numbers.
DONE	OUTPUT	BOOL	I, Q, M,D, L	DONE status parameter: sending completed
ERROR	OUTPUT	BOOL	I, Q, M,D, L	ERROR status parameter
STATUS	OUTPUT	INT	I, Q, M,D, L	STATUS status parameter
SD_1	IN_OUT	ANY	I, Q, M,D, T, C	Pointer to the archive data. The length information is not evaluated. The following data types are permitted BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
LEN	IN_OUT	WORD	I, Q, M,D, L	Length of the data field to be sent in bytes

**Error Information**

Table 19-10 contains all the error information specific to SFB37 that can be output with the ERROR and STATUS parameters.

Table 19-10 Error Information for SFB37 “AR\_SEND”

<b>ERROR</b>	<b>STATUS (decimal)</b>	<b>Explanation</b>
0	11	Warning: new job not active because the previous job is not yet completed.
1	1	Communications problems
1	2	Negative acknowledgment, function cannot be executed
1	3	There is no logon for the specified AR_ID.
1	4	Error in the archive data pointer SD_1 involving the data length or data type
1	5	Requested reset was executed
1	7	Remote partner in wrong status
1	8	Access error occurred on the remote partner
1	10	Access to local user memory not possible (for example, access to a deleted DB)
1	12	When the SFB was called: <ul style="list-style-type: none"> <li>• an uninitialized instance DB was specified</li> <li>• an instance DB that does not belong to SFB37 was specified</li> <li>• a shared DB instead of an instance DB was specified</li> </ul>
1	20	Not enough memory.

## 19.7 Disabling Block-Related, Symbol-Related and Group Status Messages with SFC10 “DIS\_MSG”

### Description

With SFC10 “DIS\_MSG” (disable message) you can disable block-related messages generated with SFBs, symbol-related messages (SCAN) and group status messages. You select messages to be disabled using the input parameters MODE and MESGN. Calling SFC10 “DIS\_MSG” and successfully disabling a message is only possible when the disabling of a message is not already active with SFC10.

Messages that are ready to be sent when SFC10 is called but that are still in an internal buffer can no longer be disabled and are sent.

A disabled message is indicated at the ERROR and STATUS outputs of the “NOTIFY”, “ALARM”, “ALARM\_8P” and “ALARM\_8” SFBs.

You start the disabling of a message by assigning the value 1 to the REQ input parameter when SFC10 is called.

### How SFC10 Functions

Disabling is executed asynchronously, in other words it can be active throughout several SFC10 calls:

- When it is first called (REQ =1), SFC10 checks the input parameters and attempts to occupy the required system resources. If successful, the value W#16#7001 is entered in RET\_VAL, BUSY is set and disabling the message is started.  
If unsuccessful, the error information is entered in RET\_VAL and the job is terminated. BUSY must not be evaluated in this case.
- If there are further calls in the meantime, the value W#16#7002 is entered in RET\_VAL (job still being executed by the CPU) and BUSY is set. Further calls do not affect the current job.
- The last time the SFB is called, the value W#16#0000 is entered in RET\_VAL if no error occurred. BUSY then has the value 0.  
If an error occurred, the error information is entered in RET\_VAL and BUSY must not be evaluated.  
Table 2-4 in Chapter 2 provides you with an overview of the relationship described above.

## Parameters

Table 19-11 Parameters for SFC10 “DIS\_MSG”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M,D, L, constant	REQ = 1: trigger disable
MODE	INPUT	BYTE	I, Q, M,D, L, constant	Parameter for selecting the messages to be disabled, see Table 19-12
MESGN	INPUT	DWORD	I, Q, M,D, L, constant	Message number only relevant when MODE is set to 5, 6, 7. This allows a single message to be disabled.
RET_VAL	OUTPUT	INT	I, Q, M,D, L	Error information
BUSY	OUTPUT	BOOL	I, Q, M,D, L	BUSY = 1: disable has not yet been canceled.

## MODE Input Parameter

The table below shows the permitted values for the MODE input parameter:

Table 19-12 Values Permitted for MODE

Value	Meaning
0	All block-related, all symbol-related and all group status messages of the CPU generated with SFBs
1	All block-related messages of the CPU generated with SFBs, in other words all messages generated by the “NOTIFY”, “ALARM”, “ALARM_8P” and “ALARM_8” SFBs
2	All group status messages of the CPU
3	All symbol-related messages of the CPU (SCAN)
5	Single message of the “symbol-related messages” class
6	Single message of the “block-related messages” class
7	Single message of the “group status messages” class

## Error Information

Table 19-13 Error Information Specific to SFC10 “DIS\_MSG”

Error Code (W#16#...)	Explanation
0000	Disabling was terminated without an error.
7000	REQ = 0 at first call: disabling was not activated.
7001	REQ = 1 at first call: disabling was triggered.
7002	Further call: disabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The message number is outside the permitted range of values.
8084	There is no logon for the message(s) specified with MODE and possibly MESGN.
80C3	The message(s) to be disabled in MODE and possibly MESGN, cannot be disabled at present – SFC10 is already disabling messages.

## 19.8 Enabling Block-Related, Symbol-Related and Group Status Messages with SFC9 “EN\_MSG”

### Description

With SFC9 “EN\_MSG” (enable message), you can enable block-related, symbol-related and group status messages that were previously disabled. You disabled the messages either at a display device or using SFC10 “DIS\_MSG”.

You specify the messages to be enabled using the MODE and MESGN input parameters. Successful enabling of messages with SFC9 “EN\_MSG” is only possible when SFC9 is not already actively enabling messages.

You start the enabling function by assigning the value 1 to the REQ input parameter of SFC9.

### How SFC9 Functions

Enabling is executed asynchronously, in other words it can be active throughout several SFC9 calls:

- When it is first called (REQ =1), SFC9 checks the input parameters and attempts to occupy the required system resources. If successful, the value W#16#7001 is entered in RET\_VAL, BUSY is set and enabling the message is started.  
If unsuccessful, the error information is entered in RET\_VAL and the job is terminated. BUSY must not be evaluated in this case.
- If there are further calls in the meantime, the value W#16#7002 is entered in RET\_VAL (job still being executed by the CPU) and BUSY is set. Further calls do not affect the current job.
- The last time the SFB is called, the value W#16#0000 is entered in RET\_VAL if no error occurred. BUSY then has the value 0.  
If an error occurred, the error information is entered in RET\_VAL and BUSY must not be evaluated.  
Table 2-4 in Chapter 2 provides you with an overview of the relationship described above.

### Parameters

Table 19-14 Parameters for SFC9 “EN\_MSG”

Parameter	Declaration	Data Type	Memory Area	Description
REQ	INPUT	BOOL	I, Q, M,D, L, constant	REQ = 1: trigger enable
MODE	INPUT	BYTE	I, Q, M,D, L, constant	Parameter for selecting the messages to be enabled, see Table 19-12
MESGN	INPUT	DWORD	I, Q, M,D, L, constant	Message number only relevant when MODE is set to 5, 6, 7. This allows a single message to be enabled.
RET_VAL	OUTPUT	INT	I, Q, M,D, L	Error information
BUSY	OUTPUT	BOOL	I, Q, M,D, L	BUSY = 1: the enable has not yet been canceled.

**MODE Input Parameter**

The following table shows the permitted values for the MODE input parameter.

Table 19-15 Values Permitted for MODE

Value	Meaning
0	All block-related, all symbol-related and all group status messages of the CPU generated with SFBs
1	All block-related messages of the CPU generated with SFBs, in other words all messages generated by the “NOTIFY”, “ALARM”, “ALARM_8P” and “ALARM_8” SFBs
2	All group status messages of the CPU
3	All symbol-related messages of the CPU (SCAN)
5	Single message of the “symbol-related messages” class
6	Single message of the “block-related messages” class
7	Single message of the “group status messages” class

**Error Information**

Table 19-16 Error Information Specific to SFC9 “EN\_MSG”

Error Code (W#16#...)	Explanation
0000	Enabling was terminated without an error.
7000	REQ = 0 at first call: enabling was not activated.
7001	REQ = 1 at first call: enabling was triggered.
7002	Further call: enabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The message number is outside the permitted range of values.
8084	There is no logon for the message(s) specified with MODE and possibly MESGN.
80C3	The message(s) to be enabled as specified in MODE and possibly MESGN, cannot be enabled at present since SFC10 is already enabling messages.

## 19.9 Startup Behavior of the SFBs for Generating Block-Related Messages

<b>Complete Restart</b>	During a complete restart, the SFBs for generating block-related messages are set to the NO_INIT status. The actual parameters stored in the instance DBs are unchanged.
<b>Restart</b>	During a restart, the SFBs for generating block-related messages behave like user function blocks that are capable of resuming execution. They continue from the point of interruption.
<b>Memory Reset</b>	A memory reset always causes the termination of all connections so that no station is logged on for messages. The user program is deleted. If you have inserted a FLASH card, the program sections relevant to execution are loaded on the CPU again from the card and the CPU executes a complete restart.

## 19.10 How the SFBs for Generating Block-Related Messages React to Problems

### Connection Breakdown

The connections assigned to the SFB instances are monitored for breakdown. If a connection breaks down, the stations involved are removed from the internal CPU list of stations logged on for block-related messages. Any messages pending for these stations are deleted.

If other stations are still logged on following a connection breakdown, they continue to receive messages. The SFBs only stop sending messages when there are no more connections to any logged on stations. The ERROR and STATUS output parameters indicate this situation (ERROR = 1, STATUS = 1).

### Error Interface to the User Program

If an error occurs during the execution of an SFB for generating block-related messages, the SFB changes to the ERROR or ERROR\_E status. At the same time, the ERROR output parameter is set to 1 and the STATUS output parameter has the corresponding error identifier. You can evaluate this error information in your program.

Examples of possible errors:

- Sending not possible due to lack of resources
- Error accessing one of the signals to be monitored



## 19.11 Introduction to Generating Block-Related Messages with SFCs

### SFCs for Generating Block-Related Messages

You can generate a block-related message with the following SFCs:

- SFC17 “ALARM\_SQ”
- SFC18 “ALARM\_S”

These SFCs have the following properties:

- The messages sent by SFC17 “ALARM\_SQ” when the signal state is 1 can be acknowledged at a logged on display device. The messages of SFC18 “ALARM\_S” are always implicitly acknowledged.
- It is not a detected edge change that generates a message but rather each SFC call. For more detailed information refer to Section 19.12.
- The SD\_1 accompanying value is queried and assigned to the message when the SFC is called.

### SFC19 “ALARM\_SC”

Using SFC19 “ALARM\_SC” you can query the following:

- The acknowledgment status of the last “entering state message” and the signal state at the last SFC17 call or
- The signal state at the last SFC18 call

### Logging On Display Devices

The SFCs for generating block-related messages only send a message when they are called if at least one display device has logged on for block-related messages.

### Message Storage

To avoid messages being lost when there is a lot of traffic on the communications system, the SFCs 17 and 18 can both buffer two messages. If, however, messages are lost, you are informed in RET\_VAL. The logged on display devices are informed of this the next time a message can be sent.

### Message Acknowledgment with SFC17 “ALARM\_SQ”

If you have acknowledged an “entering event message” at a display device, this acknowledgment information is first sent to the CPU where the message originated. This then distributes the acknowledgment information to all stations logged on for this purpose.

### Disabling and Enabling Messages

Block-related messages generated with SFC17 “ALARM\_SQ” or SFC18 “ALARM\_S” cannot be disabled and then enabled again.

### **Message Update**

At a display device, you can use a message update to read out the current signal and acknowledgment status. During the update, all the logged on stations continue to receive the messages for which they logged on.

### **Changes in Your Program**

---

#### **Note**

When you download a block that is already on the CPU using SFC17/SFC18 calls, it is possible that the previous block has sent an entering state message but that the new block does not send a corresponding leaving state message. This means that the message remains in the internal message memory of the CPU. This situation can also occur when you delete blocks with SFC17/SFC18.

You can remove such messages from the internal message memory of the CPU by changing the CPU to STOP and then going through a complete re-start.

---

## 19.12 Generating Acknowledgable Block-Related Messages with SFC 17 “ALARM\_SQ” and Permanently Acknowledged Block-Related Messages with SFC18 “ALARM\_S”

### Description

Each time they are called, SFC17 “ALARM\_SQ” and SFC18 “ALARM\_S” generate a message to which you can add accompanying values. The message is sent to all stations that have logged on for the message.

SFC17 and SFC18 provide you with a simple mechanism for sending messages. You must make sure that you only call SFC17 or SFC18 when the value of the triggering signal SIG is inverted compared with the last call. If this is not the case, this is indicated in RET\_VAL and no message is sent. The very first time that SFC17 or SFC18 is called, you must make sure that the SIG input has the value 1. Otherwise RET\_VAL contains error information and no message will be sent.

---

### Note

Call SFC17 and SFC18 in an FB to which you have previously assigned suitable system attributes!  
For more detailed information about assigning system attributes to blocks, refer to [/232/](#) and [/233/](#).

---

### Acknowledging Messages

You can acknowledge messages sent by SFC17 “ALARM\_SQ” when the state of the monitored signal is 1.

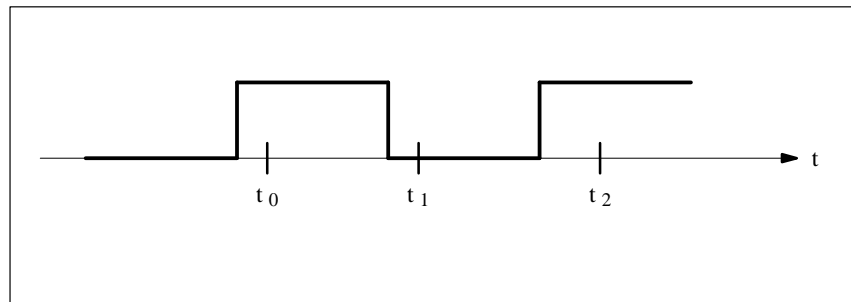
You can query the acknowledgment status of the last “entering event message” and the signal state at the last SFC call using SFC19 “ALARM\_SC”.

Messages you have sent with SFC18 “ALARM\_S” are always implicitly acknowledged. You can query the signal state at the last SFC18 call using SFC19 “ALARM\_SC”.

### Temporary Storage of Signal States

SFC17 “ALARM\_SQ” and SFC18 “ALARM\_S” occupy memory temporarily. Here, they enter among other things the last two signal states including the time stamp and accompanying value. If SFC17 or SFC18 is called at a time when the signal states of the two last “valid” SFC calls have not yet been sent (signal overflow), the current and the last signal state are discarded and an overflow ID is set in the buffer. At the next possible opportunity, the second but last signal and the overflow identifier are sent.

Example:



$t_0$ ,  $t_1$  and  $t_2$  are the points at which SFC17 or SFC18 are called. If the signal states of  $t_0$  and  $t_1$  are not sent at the time  $t_2$ , the signal states of  $t_1$  and  $t_2$  are discarded and the overflow identifier is set for the signal state of  $t_0$ .

### Instance Overflow

If the number of SFC17 or SFC18 calls is higher than the maximum number of dynamic instances, this may result in a lack of resources (instance overflow). This is indicated both by the information in RET\_VAL as well as by indications at the logged on display devices.

The maximum number of SFC17 or SFC18 calls depends on the particular CPU. You will find this information in **/70/** and **/101/**.

### Parameters

Table 19-17 Parameters for SFC17 “ALARM\_SQ” and SFC18 “ALARM\_S”

Parameter	Declaration	Data Type	Memory Area	Description
SIG	INPUT	BOOL	I, Q, M,D, L, constant	The signal to trigger a message
ID	INPUT	WORD	I, Q, M,D, L, constant	Data channel for messages: W#16#EEEE
EV_ID	INPUT	DWORD	I, Q, M,D, L, constant	Message number (0 not permitted) When assigning the message number, use the message configuration functions. This ensures the consistency of the message numbers.
SD_1	IN_OUT	ANY	I, Q, M,D, T, C	Accompanying value Maximum length: 12 bytes The following data types are permitted BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER.
RET_VAL	OUTPUT	INT	I, Q, M,D, L	Error information

## Error Information

Table 19-18 Error Information Specific to SFC17 “ALARM\_SQ” and SFC18 “ALARM\_S”

Error Code (W#16#...)	Explanation
0000	No error occurred.
0001	<ul style="list-style-type: none"> <li>The accompanying value is longer than the maximum permitted length or</li> <li>Access to the user memory is not possible (for example, access to a deleted DB). The message is sent.</li> </ul>
8081	The specified EV_ID is outside the permitted range.
8082	Loss of messages since your CPU has no more resources for generating block-related messages by SFCs.
8083	Message loss since the same signal change already exists but could not yet be sent (signal overflow).
8084	The signal that triggered the message (SIG) had the same value at the current SFC17 or SFC18 call as at the last call.
8085	No logon for the specified EV_ID
8086	An SFC call for the specified EV_ID is already being executed in a lower priority class.
8087	When SFC17 or SFC18 were first called, the message trigger signal had the value 0.
8088	The specified EV_ID is already being used by another SFC type that is currently (still) occupying memory.

### 19.13 Querying the Acknowledgment Status of the Last ALARM\_SQ Entering Event Message with SFC19 “ALARM\_SC”

#### Description

With SFC19 “ALARM\_SC” you can query the following:

- The acknowledgment status of the last ALARM\_SQ entering state message and the status of the signal that triggered the message the last time that SFC17 “ALARM\_SQ” was called
- The status of the signal that triggered the message the last time SFC18 “ALARM\_S” was called

Assuming that you assigned the message numbers during message configuration, the message or signal is referenced with a unique message number

SFC19 “ALARM\_SC” accesses the temporarily occupied memory of SFC17 or SFC18.

#### Parameters

Table 19-19 Parameters for SFC19 “ALARM\_SC”

Parameter	Declaration	Data Type	Memory Area	Description
EV_ID	INPUT	DWORD	I, Q, M,D, L, constant	Message number for the signal state at the last SFC call or the acknowledgment status of the last entering state message (only with SFC17) that you want to query.
RET_VAL	OUTPUT	INT	I, Q, M,D, L	Error information
STATE	OUTPUT	BOOL	I, Q, M,D, L	State of the signal that triggered the message at the last SFC call
Q_STATE	OUTPUT	BOOL	I, Q, M,D, L	If the specified EV_ID parameter belongs to an SFC18 call: 1 If the specified EV_ID parameter belongs to an SFC call: acknowledgment status of the last entering state message: 0: Not acknowledged 1: Acknowledged

#### Error Information

Table 19-20 Error Information Specific to SFC19 “ALARM\_SC”

Error Code (W#16#...)	Explanation
0000	No error occurred.
8081	The specified EV_ID is outside the permitted range.
8082	No memory is currently occupied for this EV_ID (possible cause: the corresponding signal state was not yet 1, or the signal state has already returned to 0).

# IEC Timers and IEC Counters

# 20

## Chapter Overview

Section	Description	Page
20.1	Generating a Pulse with SFB3 “TP”	20-2
20.2	Generating an On Delay with SFB4 “TON”	20-3
20.3	Generating an Off Delay with SFB5 “TOF”	20-4
20.4	Counting Up with SFB0 “CTU”	20-5
20.5	Counting Down with SFB1 “CTD”	20-6
20.6	Counting Up and Down with SFB2 “CTUD”	20-7

## 20.1 Generating a Pulse with SFB3 “TP”

### Description

SFB3 “TP” generates a pulse with the length PT. The timer runs only in the STARTUP and RUN modes.

A rising signal edge at input IN starts the pulse. Output Q remains set for the time PT regardless of changes in the input signal (in other words even when the IN input changes back from 0 to 1 before the time PT has expired).

The ET output provides the time for which output Q has already been set.

The maximum value of the ET output is the value of the PT input. Output ET is reset when input IN changes to 0, however, not before the time PT has expired.

SFB3 “TP” complies with the IEC 1131-3 standard.

The operating system does not reset SFB3 “TP” during a complete restart. If you want SFB3 “TP” to be initialized following a complete restart, you must call SFB3 with PT = 0 ms in OB100.

### Timing Diagram

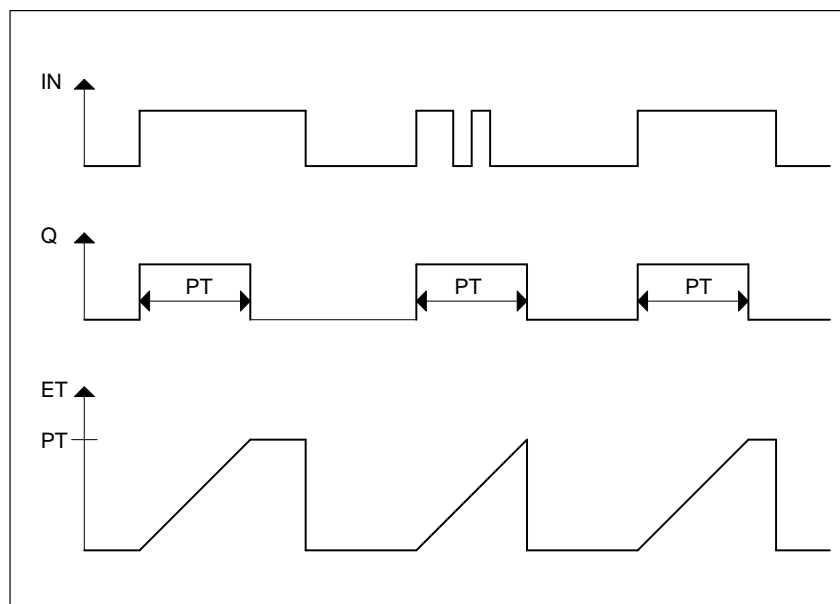


Figure 20-1

Timing Diagram for SFB3 “TP”

### Parameters

Table 20-1 Parameters for SFB3 “TP”

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Duration of the pulse. PT must be positive.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time



## 20.2 Generating an On Delay with SFB4 “TON”

### Description

SFB4 “TON” delays a rising signal edge by the time PT. The timer runs only in the STARTUP and RUN modes.

A rising edge at the IN input causes a rising edge at output Q after the time PT has expired. Q then remains set until the IN input changes to 0 again. If the IN input changes to 0 before the time PT has expired, output Q remains set to 0.

The ET output provides the time that has passed since the last rising edge at the IN input. Its maximum value is the value of the PT input. ET is reset when the IN input changes to 0.

SFB4 “TON” complies with the IEC 1131-3 standard.

The operating system does not reset SFB4 “TON” during a complete restart. If you want SFB4 “TON” to be initialized following a complete restart, you must call SFB4 with PT = 0 ms in OB100.

### Timing Diagram

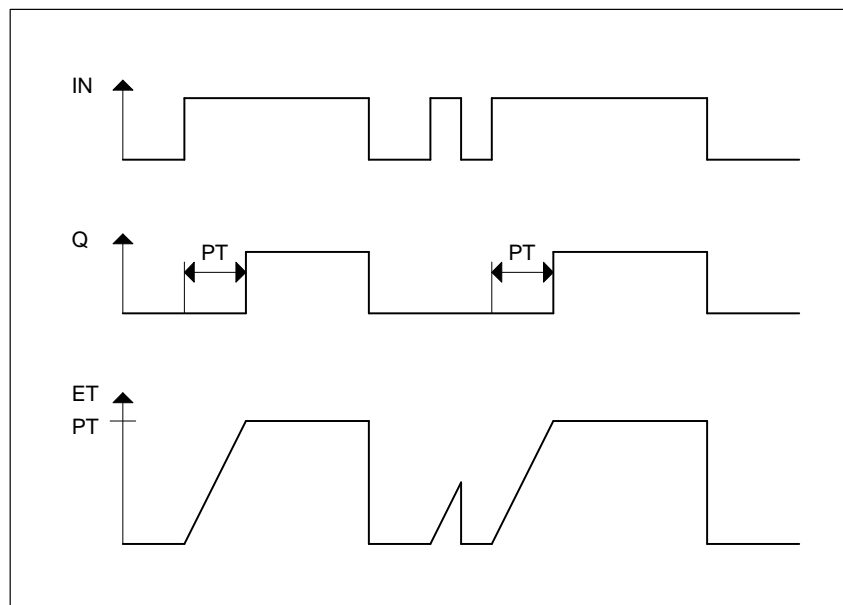


Figure 20-2 Timing Diagram for SFB4 “TON”

### Parameters

Table 20-2 Parameters for SFB4 “TON”

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Time by which the rising edge at the IN input is delayed. PT must be positive.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time

## 20.3 Generating an Off Delay with SFB5 “TOF”

### Description

SFB5 “TOF” delays a falling edge by the time PT. The timer runs only in the STARTUP and RUN modes.

A rising edge at the IN input causes a rising edge at output Q. A falling edge at the IN input causes a falling edge at output Q delayed by the time PT. If the IN input changes back to 1 before the time PT has expired, output Q remains set to 1.

The ET output provides the time that has elapsed since the last falling edge at the IN input. Its maximum value is, however, the value of the PT input.

ET is reset when the IN input changes to 1.

SFB5 “TOF” complies with the IEC 1131-3 standard.

The operating system does not reset SFB5 “TOF” during a complete restart. If you want SFB5 “TOF” to be initialized following a complete restart, you must call SFB5 with PT = 0 ms in OB100.

### Timing Diagram

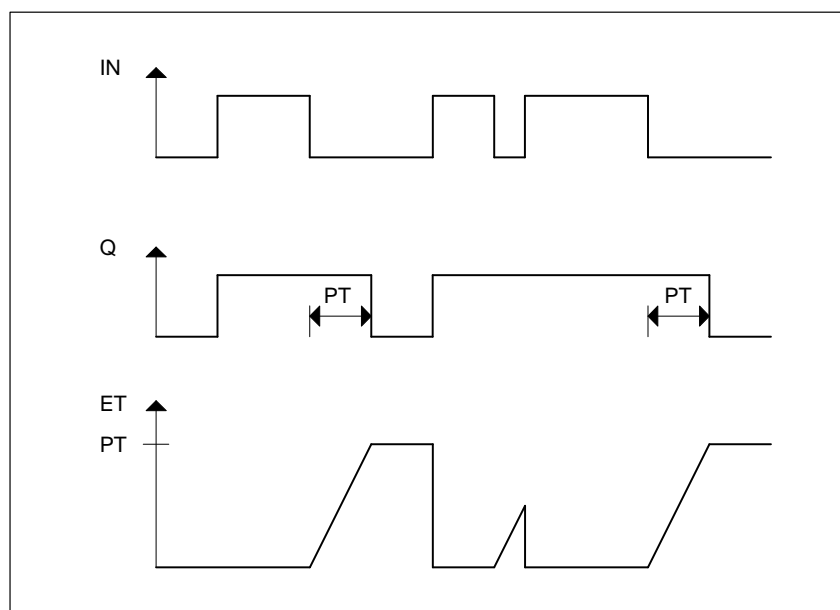


Figure 20-3

Timing Diagram for SFB5 “TOF”

### Parameters

Table 20-3 Parameters for SFB5 “TOF”

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	BOOL	I, Q, M, D, L, constant	Start input
PT	INPUT	TIME	I, Q, M, D, L, constant	Time by which the falling edge at the IN input is delayed. PT must be positive.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the time
ET	OUTPUT	TIME	I, Q, M, D, L	Expired time

## 20.4 Counting Up with SFB0 “CTU”

### Description

You can count up with SFB0 “CTU”. The counter is incremented by 1 by a rising edge at the CU input (compared with the last SFB call). If the counted value reaches the upper limit of 32767, it is no longer incremented. Each subsequent rising edge at the CU input no longer has an effect.

Signal level 1 at the R input resets the counter to the value 0 regardless of the value currently at the CU input.

The Q output indicates whether the current counted value is greater or equal to the preset value PV.

SFB0 “CTU” complies with the IEC 1131-3 standard.

The operating system does not reset SFB0 “CTU” during a complete restart. If you want SFB0 “CTU” to be initialized following a complete restart, you must call SFB0 with PT = 0 ms in OB100.

### Parameters

Table 20-4 Parameters for SFB0 “CTU”

Parameter	Declaration	Data Type	Memory Area	Description
CU	INPUT	BOOL	I, Q, M, D, L, constant	Counter input
R	INPUT	BOOL	I, Q, M, D, L, constant	Reset input R is dominant over CU.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value. Refer to parameter Q for the effect of PV.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the counter: Q has the following value <ul style="list-style-type: none"> <li>• 1, if <math>CV \geq PV</math></li> <li>• 0 otherwise</li> </ul>
CV	OUTPUT	INT	I, Q, M, D, L	Current count value (possible value: 0 to 32 767)

## 20.5 Counting Down with SFB1 “CTD”

### Description

You can count down with SFB1 “CTD”. The counter is decremented by a rising edge at the CD input (compared with the last SFB call). If the count value reaches the lower limit of –32768, it is no longer decremented. Any further rising edge at the CD input then has no further effect.

Signal level 1 at the LOAD input sets the counter to the preset value PV regardless of the value at the CD input.

The Q output indicates whether the current counted value is less than or equal to 0.

SFB1 “CTD” complies with the IEC 1131-3 standard.

The operating system does not reset SFB1 “CTD” during a complete restart. If you want SFB1 “CTD” to be initialized following a complete restart, you must call SFB1 with LOAD = 1 and PV = required initial value for CV in OB100.

### Parameters

Table 20-5 Parameters for SFB1 “CTD”

Parameter	Declaration	Data Type	Memory Area	Description
CD	INPUT	BOOL	I, Q, M, D, L, constant	Count input
LOAD	INPUT	BOOL	I, Q, M, D, L, constant	Load input. LOAD is dominant over CD.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
Q	OUTPUT	BOOL	I, Q, M, D, L	Status of the counter: Q has the following value: <ul style="list-style-type: none"> <li>• 1, if <math>CV \leq 0</math></li> <li>• 0 otherwise</li> </ul>
CV	OUTPUT	INT	I, Q, M, D, L	Current count value (possible values: –32 768 to 32 767)

## 20.6 Counting Up and Counting Down with SFB2 “CTUD”

### Description

You can count up and down with SFB2 “CTUD”. The count value is changed by a rising edge, compared with the last SFB call as follows:

- At input CU it is incremented by 1
- At input CD it is decremented by 1.

If the count value reaches the limits, the counter reacts as follows:

- The lower limit of –32768, it is no longer decremented
- The upper limit of 32767, it is no longer incremented

If there is a rising edge at both input CU and input CD in one cycle, the counter retains its current value. This reaction does not comply with the standard IEC 1131–3. In the standard, the CU input is dominant if both signals are active at the same time. This change has been proposed to the IEC.

A signal level 1 at the LOAD input presets the counter to the value PV regardless of the values at the CU and CD inputs.

The signal level 1 at the R input resets the counter to the value 0 regardless of the values at the CU, CD and LOAD inputs. The QU output indicates whether the current count value is greater than or equal to the preset value PV; the QD output indicates whether the value is less than or equal to 0.

SFB2 “CTUD” complies with the IEC 1131-3 standard.

The operating system does not reset SFB2 “CTUD” during a complete restart. If you want SFB2 “CTUD” to be initialized following a complete restart, you must call SFB2 in OB100 as follows:

- With R = 1 when using the block to count up
- With R = 0 and LOAD = 1 and PV = required initial value for CV when using the block to count down

### Parameters

Table 20-6 Parameters for SFB2 “CTUD”

Parameter	Declaration	Data Type	Memory Area	Description
CU	INPUT	BOOL	I, Q, M, D, L, constant	Count up input.
CD	INPUT	BOOL	I, Q, M, D, L, constant	Count down input
R	INPUT	BOOL	I, Q, M, D, L, constant	Reset input. R is dominant over LOAD.
LOAD	INPUT	BOOL	I, Q, M, D, L, constant	Load input. LOAD is dominant over CU and CD.
PV	INPUT	INT	I, Q, M, D, L, constant	Preset value. The counter is set to the preset value PV when the signal level at the LOAD input is 1.

Table 20-6 Parameters for SFB2 “CTUD”

Parameter	Declaration	Data Type	Memory Area	Description
QU	OUTPUT	BOOL	I, Q, M, D, L	Status of the up counter: QU has the following value <ul style="list-style-type: none"><li>• 1, if <math>CV \geq PV</math></li><li>• 0 otherwise</li></ul>
QD	OUTPUT	BOOL	I, Q, M, D, L	Status of the down counter: QD has the following value <ul style="list-style-type: none"><li>• 1, if <math>CV \leq 0</math></li><li>• 0 otherwise</li></ul>
CV	OUTPUT	INT	I, Q, M, D, L	Current count value (possible values: –32 768 to 32 767)

# IEC Functions

# 21

## Chapter Overview

Section	Description	Page
21.1	Overview	21-2
21.2	Technical Specifications for the IEC Functions	21-3
21.3	Date and Time as Complex Data Types	21-4
21.4	Date and Time-of-Day Functions: FC3, FC6, FC7, FC8, FC33, FC40, FC1, FC35, FC34	21-5
21.5	Comparing DATE_AND_TIME Variables: FC9, FC12, FC14, FC18, FC23, FC28	21-10
21.6	Comparing STRING Variables: FC10, FC13, FC15, FC19, FC24, FC29	21-13
21.7	Editing STRING Variables: FC21, FC20, FC32, FC26, FC2, FC17, FC4, FC31, FC11	21-16
21.8	Converting Data Type Formats: FC16, FC5, FC30, FC38, FC37, FC39	21-21
21.9	Editing Number Values: FC22, FC25, FC27	21-24
21.10	Binary Selection: FC36	21-26

## 21.1 Overview

### Available Functions

You can copy the following International Electrotechnical Commission (IEC) functions from the STEP 7 library STDLIBS\IEC to your program directory.

Function	Name	IEC Block Family
Combine DATE and TIME_OF_DAY to DT	D_TOD_DT FC3	Convert
Extract the DATE from DT	DT_DATE FC6	Convert
Extract the day of the week from DT	DT_DAY FC7	Convert
Extract the TIME_OF_DAY from DT	DT_TOD FC8	Convert
Data type conversion S5TIME to TIME	S5TI_TIM FC33	Convert
Data type conversion TIME to S5TIME	TIM_S5TI FC40	Convert
Data type conversion INT to STRING	I_STRNG FC16	Convert
Data type conversion DINT to STRING	DI_STRNG FC5	Convert
Data type conversion REAL to STRING	R_STRNG FC30	Convert
Data type conversion STRING to INT	STRNG_I FC38	Convert
Data type conversion STRING to DINT	STRNG_DI FC37	Convert
Data type conversion STRING to REAL	STRNG_R FC39	Convert
Compare DT for equal	EQ_DT FC9	DT
Compare DT for greater than or equal	GE_DT FC12	DT
Compare DT for greater than	GT_DT FC14	DT
Compare DT for less than or equal	LE_DT FC18	DT
Compare DT for less than	LT_DT FC23	DT
Compare DT for unequal	NE_DT FC28	DT
Compare STRING for equal	EQ_STRNG FC10	String
Compare STRING for greater than or equal	GE_STRNG FC13	String
Compare STRING for greater than	GT_STRNG FC15	String
Compare STRING for less than or equal	LE_STRNG FC19	String
Compare STRING for less than	LT_STRNG FC24	String
Compare STRING for unequal	NE_STRNG FC29	String
Length of a STRING variable	LEN FC21	String
Left part of a STRING variable	LEFT FC20	String
Right part of a STRING variable	RIGHT FC32	String
Middle part of a STRING variable	MID FC26	String
Combine two STRING variables	CONCAT FC2	String
Insert in a STRING variable	INSERT FC17	String
Delete in a STRING variable	DELETE FC4	String
Replace in a STRING variable	REPLACE FC31	String
Find in a STRING variable	FIND FC11	String
Add duration to a time	AD_DT_TM FC1	Floating-Point Math
Subtract duration from a time	SB_DT_TM FC35	Floating-Point Math
Subtract two time values	SB_DT_DT FC34	Floating-Point Math
Limit	LIMIT FC22	Floating-Point Math
Select maximum	MAX FC25	Floating-Point Math
Select minimum	MIN FC27	Floating-Point Math
Binary selection	SEL FC36	Floating-Point Math



## 21.2 Technical Specifications for the IEC Functions

### Memory Requirements

The following Table shows how much work memory and load memory are required for each IEC date and time function as well as the number of bytes of local data required for each IEC date and time function.

Name		Bytes of		Bytes of Local Data
		Work Memory	Load Memory	
D_TOD_DT	FC3	634	810	12
DT_DATE	FC6	340	466	10
DT_DAY	FC7	346	472	10
DT_TOD	FC8	114	210	6
S5TI_TIM	FC33	94	208	2
TIM_S5TI	FC40	104	208	6
I_STRNG	FC16	226	340	10
DI_STRNG	FC5	314	440	18
R_STRNG	FC30	528	684	28
STRNG_I	FC38	292	420	12
STRNG_DI	FC37	310	442	12
STRNG_R	FC39	828	1038	30
EQ_DT	FC9	96	194	2
GE_DT	FC12	174	288	4
GT_DT	FC14	192	310	4
LE_DT	FC18	168	280	4
LT_DT	FC23	192	310	4
NE_DT	FC28	96	194	2
EQ_STRNG	FC10	114	220	4
GE_STRNG	FC13	162	282	8
GT_STRNG	FC15	158	278	8
LE_STRNG	FC19	162	282	8
LT_STRNG	FC24	158	278	8
NE_STRNG	FC29	150	266	8
LEN	FC21	38	132	2
LEFT	FC20	200	320	8
RIGHT	FC32	230	350	8
MID	FC26	264	390	8
CONCAT	FC2	320	452	14
INSERT	FC17	488	644	20
DELETE	FC4	376	512	8
REPLACE	FC31	562	726	20
FIND	FC11	236	360	14
AD_DT_TM	FC1	1350	1590	22
SB_DT_TM	FC35	1356	1596	22
SB_DT_DT	FC34	992	1178	30
LIMIT	FC22	426	600	12
MAX	FC25	374	532	8
MIN	FC27	374	532	8
SEL	FC36	374	560	8

## 21.3 Date and Time as Complex Data Types

### Actual Parameters for DATE\_AND\_TIME

The DATE\_AND\_TIME data type falls into the category of complex data types, along with ARRAY, STRING, and STRUCT. The permitted memory areas for complex data types are the data block (DB) and local data (L stack) areas.

Because DATE\_AND\_TIME is a complex data type, when you use it as a formal parameter in an instruction, you must specify the actual parameter in one of the following forms:

- As a block-local symbol from the variable declaration table for a specific block
- As a symbolic name for a data block, such as “DB\_sys\_info.System\_Time,” made up of the following two parts:
  - A name defined in the symbol table for the number of the data block (for example, “DB\_sys\_info” for DB5)
  - A name defined within the data block for the DATE\_AND\_TIME element (for example, “System\_Time” for a variable of data type DATE\_AND\_TIME contained in DB5)

You cannot use constants as actual parameters for formal parameters of the complex data types, including DATE\_AND\_TIME. Nor can you transfer absolute addresses as actual parameters to DATE\_AND\_TIME.

## 21.4 Date and Time-of-Day Functions: FC3, FC6, FC7, FC8, FC33, FC40, FC1, FC35, FC34

**Description of FC3** The function FC3 combines the data formats DATE and TIME\_OF\_DAY (TOD) and converts these formats to the data type format DATE\_AND\_TIME (DT). The input value IN1 must be between the limits DATE#1990-01-01 and DATE#2089-12-31 (this value is not checked). The function does not report any errors.

### Parameters

Table 21-1 Parameters for IEC Function FC3: Combine the Data Formats DATE and TIME\_OF\_DAY to the Data Format DATE\_AND\_TIME

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	DATE	I, Q, M, D, L, constant	Input variable in the DATE format
IN2	INPUT	TIME_OF_DAY	I, Q, M, D, L, constant	Input variable in the TOD format
RET_VAL	OUTPUT	DATE_AND_TIME	D, L	Return value in the DT format

You can assign only a symbolically defined variable for the return value.

**Description of FC6** The function FC6 extracts the data type format DATE from the format DATE\_AND\_TIME. The DATE value must be between the limits DATE#1990-1-1 and DATE#2089-12-31. The function does not report any errors.

### Parameters

Table 21-2 Parameters for IEC Function FC6: Extract the DATE Data Format from the DATE\_AND\_TIME Format

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	DATE_AND_TIME	D, L	Input variable in the DT format
RET_VAL	OUTPUT	DATE	I, Q, M, D, L	Return value in the DATE format

You can assign only a symbolically defined variable for the input value.

**Description of FC7**

The IEC function FC7 extracts the day of the week from the format DATE\_AND\_TIME.

The day of the week is supplied in the INT data format (from 1 to 7):

- 1 Sunday
- 2 Monday
- 3 Tuesday
- 4 Wednesday
- 5 Thursday
- 6 Friday
- 7 Saturday

The function does not report any errors.

**Parameters**

Table 21-3 Parameters for IEC Function FC7: Extract the Day of the Week from the DATE\_AND\_TIME Format

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	DATE_AND_TIME	D, L	Input variable in the DT format
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Return value in the INT format

You can assign only a symbolically defined variable for the input value.

**Description of FC8**

The IEC function FC8 extracts the TIME\_OF\_DAY data format from the DATE\_AND\_TIME format. The function does not report any errors.

**Parameters**

Table 21-4 Parameters for IEC Function FC8: Extract the TIME\_OF\_DAY Data Format from the DATE\_AND\_TIME Format

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	DATE_AND_TIME	D, L	Input variable in the DT format
RET_VAL	OUTPUT	TIME_OF_DAY	I, Q, M, D, L	Return value in the TOD format

You can assign only a symbolically defined variable for the input value.

**Description of FC33**

The IEC function FC33 converts the data type format S5TIME to the TIME format. If the result of the conversion is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-5 Parameters for IEC Function FC33: Change from the S5TIME Format to the TIME Format

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	S5TIME	I, Q, M, D, L, constant	Input variable in the S5TIME format
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	Return value in the TIME format

**Description of FC40**

The function FC40 converts the data type format TIME to the S5TIME format. The value is rounded down during conversion. If the input parameter is greater than the S5TIME format allows (greater than TIME#02:46:30.000), the result S5TIME#999.3 is output and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-6 Parameters for IEC Function FC40: Change from the TIME Format to the S5TIME Format

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	TIME	I, Q, M, D, L, constant	Input variable in the TIME format
RET_VAL	OUTPUT	S5TIME	I, Q, M, D, L	Return value in the S5TIME format

**Description of FC1**

The function FC1 adds a duration (format TIME) to a time (format DT) and provides a new time (format DT) as the result. The time (parameter T) must be in the range from DT#1990-01-01-00:00:00.000 to DT#2089-12-31-23:59:59.999. The function does not run an input check. If the result of the addition is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-7 Parameters for IEC Function FC1: Add Duration to a Time

Parameter	Declaration	Data Type	Memory Area	Description
T	INPUT	DT	I, Q, M, D, L	Time in format DT
D	INPUT	TIME	I, Q, M, D, L, constant	Duration in format TIME
RET_VAL	OUTPUT	DT	I, Q, M, D, L	Sum in format DT

You can assign only a symbolically defined variable for the input parameter T and the output parameter.

**Description of FC35**

The function FC35 subtracts a duration (format TIME) from a time (format DT) and provides a new time (format DT) as the result. The time (parameter T) must be between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The function does not run an input check. If the result of the subtraction is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-8 Parameters for IEC Function FC35: Subtract a Duration from a Time

Parameter	Declaration	Data Type	Memory Area	Description
T	INPUT	DT	I, Q, M, D, L	Time in format DT
D	INPUT	TIME	I, Q, M, D, L, constant	Duration in format TIME
RET_VAL	OUTPUT	DT	I, Q, M, D, L	Difference in format DT

You can assign only a symbolically defined variable for the input parameter T and the output parameter.

## Description of FC34

The function FC34 subtracts two time values (format DT) and provides a duration (format TIME) as the result. The times must be in the range from DT#1990-01-01-00:00:00.000 to DT#2089-12-31-23:59:59.999. The function does not run an input check. If the first time (parameter T1) is greater (more recent) than the second (parameter T2), the result is positive; if the first time is less (less recent) than the second, the result is negative. If the result of the subtraction is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

## Parameters

Table 21-9 Parameters for IEC Function FC34: Subtract two Time Values

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DT	I, Q, M, D, L	First time in format DT
DT2	INPUT	DT	I, Q, M, D, L	Second time in format DT
RET_VAL	OUTPUT	TIME	I, Q, M, D, L	Difference in format TIME

You can assign only a symbolically defined variable for the input parameters.

## 21.5 Comparing DATE\_AND\_TIME Variables: FC9, FC12, FC14, FC18, FC23, FC28

**Description of FC9** The function FC9 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if they are equal and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is the same as the time at parameter DT2. The function does not report any errors.

### Parameters

Table 21-10 Parameters for IEC Function FC9: Compare DATE\_AND\_TIME for equal

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

**Description of FC12** The function FC12 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if one is greater or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is greater (more recent) than the time at parameter DT2 or if both times are the same. The function does not report any errors.

### Parameters

Table 21-11 Parameters for IEC Function FC12: Compare DATE\_AND\_TIME for greater than or equal

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.



**Description of FC14**

The function FC14 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if one is greater than the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is greater (more recent) than the time at parameter DT2. The function does not report any errors.

**Parameters**

Table 21-12 Parameters for IEC Function FC14: Compare DATE\_AND\_TIME for greater than

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

**Description of FC18**

The function FC18 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if one is less than or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is less (less recent) than the time at parameter DT2 or if both times are the same. The function does not report any errors.

**Parameters**

Table 21-13 Parameters for IEC Function FC18: Compare DATE\_AND\_TIME for less than or equal

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

**Description of FC23**

The function FC23 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if one is less than the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is less (less recent) than the time at parameter DT2. The function does not report any errors.

**Parameters**

Table 21-14 Parameters for IEC Function FC23: Compare DATE\_AND\_TIME for less than

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

**Description of FC28**

The function FC28 compares the contents of two variables in the data type format DATE\_AND\_TIME to find out if they are unequal and outputs the result of the comparison as a return value. The return value has the signal state “1” if the time at parameter DT1 is not equal to the time at parameter DT2. The function does not report any errors.

**Parameters**

Table 21-15 Parameters for IEC Function FC28: Compare DATE\_AND\_TIME for unequal

Parameter	Declaration	Data Type	Memory Area	Description
DT1	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
DT2	INPUT	DATE_AND_TIME	D, L	Input variable in format DT
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

## 21.6 Comparing STRING Variables: FC10, FC13, FC15, FC19, FC24, FC29

### Description of FC10

The function FC10 compares the contents of two variables in the data type format STRING to find out if they are equal and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is the same as the string at parameter S2. The function does not report any errors.

### Parameters

Table 21-16 Parameters for IEC Function FC10: Compare STRING for equal

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

### Description of FC13

The function FC13 compares the contents of two variables in the data type format STRING to find out if the first is greater than or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is greater than or equal to the string at parameter S2. The characters are compared by their ASCII code (for example, ‘a’ is greater than ‘A’), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string counts as greater than the shorter. The function does not report any errors.

### Parameters

Table 21-17 Parameters for IEC Function FC13: Compare STRING for greater than or equal

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

### Description of FC15

The function FC15 compares the contents of two variables in the data type format STRING to find out if the first is greater than the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is greater than the string at parameter S2. The characters are compared by their ASCII code (for example, ‘a’ is greater than ‘A’), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string counts as greater than the shorter. The function does not report any errors.

### Parameters

Table 21-18 Parameters for IEC Function FC15: Compare STRING for greater than

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

### Description of FC19

The function FC19 compares the contents of two variables in the data type format STRING to find out if the first is less than or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is less than or equal to the string at parameter S2. The characters are compared by their ASCII code (for example, ‘a’ is less than ‘A’), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string counts as less than the longer. The function does not report any errors.

### Parameters

Table 21-19 Parameters for IEC Function FC19: Compare STRING for less than or equal

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

### Description of FC24

The function FC24 compares the contents of two variables in the data type format STRING to find out if the first is less than the other and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is less than the string at parameter S2. The characters are compared by their ASCII code (for example, ‘a’ is less than ‘A’), starting from the left. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string counts as less than the longer. The function does not report any errors.

### Parameters

Table 21-20 Parameters for IEC Function FC24: Compare STRING for less than

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

### Description of FC29

The function FC29 compares the contents of two variables in the data type format STRING to find out if they are unequal and outputs the result of the comparison as a return value. The return value has the signal state “1” if the string at parameter S1 is not equal to the string at parameter S2. The function does not report any errors.

### Parameters

Table 21-21 Parameters for IEC Function FC29: Compare STRING for unequal

Parameter	Declaration	Data Type	Memory Area	Description
S1	INPUT	STRING	D, L	Input variable in the STRING format
S2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	BOOL	I, Q, M, D, L	Result of comparison

You can assign only a symbolically defined variable for the input parameters.

## 21.7 Editing STRING Variables: FC21, FC20, FC32, FC26, FC2, FC17, FC4, FC31, FC11

### Description of FC21

A STRING variable contains two lengths: the maximum length (this is given in square brackets when the variables are being defined) and the current length (this is the number of currently valid characters). The current length must be less than or equal to the maximum length. The number of bytes occupied by a string is 2 greater than the maximum length.

The function FC21 outputs the current length of a string (number of valid characters) as a return value. A blank string ( ' ') has the length zero. The maximum length is 254. The function does not report any errors.

### Parameters

Table 21-22 Parameters for IEC Function FC21: Length of a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
S	INPUT	STRING	D, L	STRING variable
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Number of valid characters

You can assign only a symbolically defined variable for the input parameter.

### Description of FC20

The function FC20 provides the first L characters of a string (where L stands for a number). If L is greater than the current length of the STRING variables, the input value is returned. With L = 0 and with a blank string as the input value, a blank string is returned. If L is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".

### Parameters

Table 21-23 Parameters for IEC Function FC20: Left Part of a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	STRING	D, L	Input variable in the STRING format
L	INPUT	INT	I, Q, M, D, L, constant	Length of the left string
RET_VAL	OUTPUT	STRING	D, L	Output variable in the STRING format

You can assign only a symbolically defined variable for the parameter IN and the return value.

**Description of FC32**

The function FC32 provides the last L characters of a string (where L stands for a number). If L is greater than the current length of the STRING variables, the input value is returned. With L = 0 and with a blank string as the input value, a blank string is returned. If L is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-24 Parameters for IEC Function FC32: Right Part of a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	STRING	D, L	Input variable in the STRING format
L	INPUT	INT	I, Q, M, D, L, constant	Length of the right string
RET_VAL	OUTPUT	STRING	D, L	Output variable in the STRING format

You can assign only a symbolically defined variable for the parameter IN and the return value.

**Description of FC26**

The function FC26 provides the middle part of a string (L characters from the character P inclusive). If the sum of L and (P-1) exceeds the current length of the STRING variables, a string is returned from the character P to the end of the input value. In all other cases (P is outside the current length, P and/or L are equal to zero or negative), a blank string is returned and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-25 Parameters for IEC Function FC26: Middle Part of a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	STRING	D, L	Input variable in the STRING format
L	INPUT	INT	I, Q, M, D, L, constant	Length of the middle part of the string
P	INPUT	INT	I, Q, M, D, L, constant	Position of first character
RET_VAL	OUTPUT	STRING	D, L	Output variable in the STRING format

You can assign only a symbolically defined variable for the parameter IN and the return value.

**Description of FC2**

The function FC2 concatenates two STRING variables together to form one string. If the resulting string is longer than the variable given at the output parameter, the result string is limited to the maximum set length and the binary result (BR) bit of the status word set to “0”.

**Parameters**

Table 21-26 Parameters for IEC Function FC2: Combine two STRING Variables

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	STRING	D, L	Input variable in the STRING format
IN2	INPUT	STRING	D, L	Input variable in the STRING format
RET_VAL	OUTPUT	STRING	D, L	Combined string

You can assign only a symbolically defined variable for the parameters.

**Description of FC17**

The function FC17 inserts a string at parameter IN2 into the string at parameter IN1 after the character at position P. If P equals zero, the second string is inserted before the first string. If P is greater than the current length of the first string, the second string is appended to the first. If P is negative, a blank string is output and the binary result (BR) bit of the status word is set to “0”. The binary result bit is also set to “0” if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

**Parameters**

Table 21-27 Parameters for IEC Function FC17: Insert in a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	STRING	D, L	STRING variable to be inserted into
IN2	INPUT	STRING	D, L	STRING variable to be inserted
P	INPUT	INT	I, Q, M, D, L, constant	Insert position
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the input parameters IN1 and IN2 and the output parameter.



**Description of FC4** The function FC4 deletes a number of characters (L) from the character at position P (inclusive) in a string. If L and/or P are equal to zero or if P is greater than the current length of the input string, the input string is returned. If the sum of L and P is greater than the input string, the string is deleted up to the end. If L and/or P are negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".

### Parameters

Table 21-28 Parameters for IEC Function FC4: Delete in a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	STRING	D, L	STRING variable to be deleted in
L	INPUT	INT	I, Q, M, D, L, constant	Number of characters to be deleted
P	INPUT	INT	I, Q, M, D, L, constant	Position of first character to be deleted
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the input parameter IN and the output parameter.

**Description of FC31** The function FC31 replaces a number of characters (L) of the first string (IN1) from the character at position P (inclusive) with the second string (IN2). If L is equal to zero, the first string is returned. If P is equal to zero or one, the string is replaced from the first character (inclusive). If P is outside the first string, the second string is appended to the first string. If L and/or P is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0". The binary result bit is also set to "0" if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

### Parameters

Table 21-29 Parameters for IEC Function FC31: Replace in a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	STRING	D, L	STRING variable to be replaced in
IN2	INPUT	STRING	D, L	STRING variable to be inserted
L	INPUT	INT	I, Q, M, D, L, constant	Number of characters to be replaced
P	INPUT	INT	I, Q, M, D, L, constant	Position of first character to be replaced
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the input parameters IN1 and IN2 and the output parameter.

**Description of FC11**

The function FC11 provides the position of the second string (IN2) within the first string (IN1). The search starts on the left; the first occurrence of the string is reported. If the second string is not found in the first, zero is returned. The function does not report any errors.

**Parameters**

Table 21-30 Parameters for IEC Function FC11: Find in a STRING Variable

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	STRING	D, L	STRING variable to be searched in
IN2	INPUT	STRING	D, L	STRING variable to be found
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Position of the string found

You can assign only a symbolically defined variable for the input parameters IN1 and IN2.

## 21.8 Converting Data Type Formats: FC16, FC5, FC30, FC38, FC37, FC39

### Description of FC16

The function FC16 converts a variable in the INT data type format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and the binary result (BR) bit of the status word is set to "0".

### Parameters

Table 21-31 Parameters for IEC Function FC16: Data Type Conversion INT to STRING

Parameter	Declaration	Data Type	Memory Area	Description
I	INPUT	INT	I, Q, M, D, L, constant	Input value
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the output parameter.

### Description of FC5

The function FC5 converts a variable in the DINT data type format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and the binary result (BR) bit of the status word is set to "0".

### Parameters

Table 21-32 Parameters for IEC Function FC5: Data Type Conversion DINT to STRING

Parameter	Declaration	Data Type	Memory Area	Description
I	INPUT	DINT	I, Q, M, D, L, constant	Input value
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the output parameter.

**Description of FC30**

The function FC30 converts a variable in the REAL data type format to a string. The string is shown with 14 digits:

$\pm v.nnnnnnnE\pm xx$      $\pm$  Sign

v 1 digit before the decimal point

n 7 digits after the decimal point

x 2 exponential digits

If the variable given at the return parameter is too short or if no valid floating-point number is given at parameter IN, no conversion takes place and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-33 Parameters for IEC Function FC30: Data Type Conversion REAL to STRING

Parameter	Declaration	Data Type	Memory Area	Description
IN	INPUT	REAL	I, Q, M, D, L, constant	Input value
RET_VAL	OUTPUT	STRING	D, L	Result string

You can assign only a symbolically defined variable for the output parameter.

**Description of FC38**

The function FC38 converts a string to a variable in the INT data type format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 6, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the INT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-34 Parameters for IEC Function FC38: Data Type Conversion STRING to INT

Parameter	Declaration	Data Type	Memory Area	Description
S	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Result

You can assign only a symbolically defined variable for the input parameter.

**Description of FC37**

The function FC37 converts a string to a variable in the DINT data type format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 11, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the DINT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-35 Parameters for IEC Function FC37: Data Type Conversion STRING to DINT

Parameter	Declaration	Data Type	Memory Area	Description
S	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	DINT	I, Q, M, D, L	Result

You can assign only a symbolically defined variable for the input parameter.

**Description of FC39**

The function FC39 converts a string to a variable in the REAL data type format. The string must have the following format:

$\pm v.nnnnnnnE\pm xx$      $\pm$  Sign

v 1 digit before the decimal point

n 7 digits after the decimal point

x 2 exponential digits

If the length of the string is less than 14, or if it is not structured as shown above, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the REAL range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

**Parameters**

Table 21-36 Parameters for IEC Function FC39: Data Type Conversion STRING to REAL

Parameter	Declaration	Data Type	Memory Area	Description
S	INPUT	STRING	D, L	Input string
RET_VAL	OUTPUT	REAL	I, Q, M, D, L	Result

You can assign only a symbolically defined variable for the input parameter.

## 21.9 Editing Number Values: FC22, FC25, FC27

### Description of FC22

The function FC22 limits the number value of a variable to selectable limit values. Variables of the data types INT, DINT, and REAL are permitted as input values. All variables with parameters assigned must be of the same data type. The lower limit value (parameter MN) must not exceed the upper limit value (parameter MX).

The output value remains unchanged and the binary result (BR) bit of the status word is set to “0” if any of the following are true:

- A variable with parameters assigned has an illegal data type
- All variables with parameters assigned do not have the same data type
- The lower limit value is greater than the upper limit value
- A REAL variable does not represent a valid floating-point number.

### Parameters

Table 21-37 Parameters for IEC Function FC22: Limit

Parameter	Declaration	Data Type	Memory Area	Description
MN	INPUT	ANY	I, Q, M, D, L	Lower limit
IN	INPUT	ANY	I, Q, M, D, L	Input variable
MX	INPUT	ANY	I, Q, M, D, L	Upper limit
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Limited output variable

### Description of FC25

The function FC25 selects the largest of three numeric variable values. Variables of the data types INT, DINT, and REAL are permitted as input values. All variables with parameters assigned must be of the same data type.

The output value remains unchanged and the binary result (BR) bit of the status word is set to “0” if any of the following are true:

- A variable with parameters assigned has an illegal data type
- All variables with parameters assigned do not have the same data type
- A REAL variable does not represent a valid floating-point number.

## Parameters

Table 21-38 Parameters for IEC Function FC25: Select Maximum

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	ANY	I, Q, M, D, L	First input value
IN2	INPUT	ANY	I, Q, M, D, L	Second input value
IN3	INPUT	ANY	I, Q, M, D, L	Third input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Highest of the input values

## Description of FC27

The function FC27 selects the lowest of three numerical variable values. Variables of the data types INT, DINT, and REAL are permitted as input values. All variables with parameters assigned must be of the same data type.

The output value remains unchanged and the binary result (BR) bit of the status word is set to “0” if any of the following are true:

- A variable with parameters assigned has an illegal data type
- All variables with parameters assigned do not have the same data type
- A REAL variable does not represent a valid floating-point number.

## Parameters

Table 21-39 Parameters for IEC Function FC27: Select Minimum

Parameter	Declaration	Data Type	Memory Area	Description
IN1	INPUT	ANY	I, Q, M, D, L	First input value
IN2	INPUT	ANY	I, Q, M, D, L	Second input value
IN3	INPUT	ANY	I, Q, M, D, L	Third input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Smallest of the input values

## 21.10 Binary Selection: FC36

### Description of FC36

The function FC36 selects one of two variable values depending on a switch (parameter G). Variables with all data types which correspond to the data width bit, byte, word, and double word (not data types DT and STRING) are permitted as input values at the parameters IN0 and IN1. Both input variables and the output variable must be of the same data type.

The output value remains unchanged and the binary result (BR) bit of the status word is set to "0" if any of the following are true:

- A variable with parameters assigned has an illegal data type
- All variables with parameters assigned do not have the same data type
- A REAL variable does not represent a valid floating-point number.

### Parameters

Table 21-40 Parameters for IEC Function FC36: Binary Selection

Parameter	Declaration	Data Type	Memory Area	Description
G	INPUT	BOOL	I, Q, M, D, L, constant	Selection switch
IN0	INPUT	ANY	I, Q, M, D, L	First input value
IN1	INPUT	ANY	I, Q, M, D, L	Second input value
RET_VAL	OUTPUT	ANY	I, Q, M, D, L	Selected input value



# SFBs for Integrated Control

# 22

## Chapter Overview

Section	Description	Page
22.1	Continuous Control with SFB41 “CONT_C”	22-4
22.2	Step Control with SFB42 “CONT_S”	22-11
22.3	Pulse Generation with SFB43 “PULSEGEN”	22-17
22.4	Example of the PULSEGEN Block	22-27

---

### Note

The SFBs described in this Chapter are only available on the CPU 314 IFM.

---

## **Concept of Integrated Control**

The SFBs described in this chapter are called SFBs for integrated control because they are integrated in the operating system of the CPU 314 IFM. They consist of the blocks for a continuous controller (CONT\_C) and for a step controller (CONT\_S) and the SFB for pulse duration modulation (PULSEGEN).

The control blocks implement a software controller in which the block contains all the functions of the controller. The data required for cyclic calculation are located in data blocks (instance DBs). The SFBs can therefore call them more than once.

The SFB PULSEGEN is used in conjunction with SFB CONT\_C to obtain a controller with a pulse output for proportional actuators.

## **Basic Functions**

A controller implemented with the SFBs consists of a series of subfunctions that you can activate and deactivate. In addition to the actual controller with its PID algorithm, functions are also available for influencing the setpoint and process variable and for influencing the calculated manipulated variable.

## **Applications**

A controller implemented using the controller blocks is not restricted in terms of possible applications. The control performance and the speed of processing only depend on the performance of the CPU being used.

With any given CPU, a compromise must be made between the number of controllers and the frequency at which the individual controllers must be processed. The faster the connected control loops, in other words, the more often that the manipulated variables must be calculated per time unit, the lower the number of controllers that can be installed.

There are no restrictions in terms of the type of processes that can be controlled. Both slow processes (temperatures, levels, etc.) and extremely fast processes (flow rates, motor speed, etc.) can be controlled.

## **Analysis of the Process**

The static response (gain) and the dynamic characteristics (time lag, dead time, reset time, etc.) of the process have a decisive influence on the design and implementation of the controller and the dimensions of its static (P action) and dynamic (I and D action) parameters.

Exact knowledge of the type and the characteristic data of the process is absolutely necessary.

## **Selecting a Controller**

The characteristics of control loops are determined by the physical properties of a process or the machinery involved and can hardly be influenced. Good control quality is therefore only possible by selecting the most suitable controller type for a process and adapting it to the time response of the process.

**Implementing a  
Controller**

Creating a controller, from the initial structuring and assignment of parameters to its call in the system program, is possible to a large extent without programming. Knowledge of STEP 7 is, however, necessary.

**Online Help**

For further information on the SFBs, refer to the online help in STEP 7.

**Further  
Information**

The integrated controller functions are a subset of the standard controller functions. For further information on the standard controller, refer to /350/.

## 22.1 Continuous Control with SFB41 “CONT\_C”

### Introduction

SFB “CONT\_C” is used on SIMATIC S7 programmable logic controllers to control technical processes with continuous input and output variables. During parameter assignment, you can activate or deactivate subfunctions of the PID controller to adapt the controller to the process.

### Application

You can use the controller as a PID fixed setpoint controller or in multi-loop controls as a cascade, blending or ratio controller. The functions of the controller are based on the PID control algorithm of the sampling controller with an analog signal, if necessary extended by including a pulse generator stage to generate pulse duration modulated output signals for two or three step controllers with proportional actuators.

### Description

Apart from the functions in the setpoint and process value branches, the SFB implements a complete PID controller with continuous manipulated variable output and the option of influencing the manipulated value manually. In the following, you will find a detailed description of the subfunctions:

#### Setpoint Branch

The setpoint is entered in floating-point format at the **SP\_INT** input.

#### Process Variable Branch

The process variable can be input in the peripheral (I/O) or floating-point format. The **CRP\_IN** function converts the **PV\_PER** peripheral value to a floating-point format of –100 to +100 % according to the following formula:

$$\text{Output of CPR\_IN} = \text{PV\_PER} * \frac{100}{27648}$$

The **PV\_NORM** function normalizes the output of **CRP\_IN** according to the following formula:

$$\text{Output of PV\_NORM} = (\text{output of CPR\_IN}) * \text{PV\_FAC} + \text{PV\_OFF}$$

**PV\_FAC** has a default of 1 and **PV\_OFF** a default of 0.

#### Error Signal

The difference between the setpoint and process variable is the error signal. To suppress a small constant oscillation due to the manipulated variable quantization (for example in pulse duration modulation with **PULSEGEN**), a dead band is applied to the error signal (**DEADBAND**). If **DEADB\_W** = 0, the dead band is switched off.

#### PID Algorithm

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and derivative (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured. Pure I and D controllers are also possible.

**Manual Value**

It is possible to switch over between a manual and an automatic mode. In the manual mode, the manipulated variable is corrected to a manually selected value. The integrator (INT) is set internally to  $LMN - LMN\_P - DISV$  and the derivative unit (DIF) to 0 and matched internally. This means that a switchover to the automatic mode does not cause any sudden change in the manipulated value.

**Manipulated Value**

The manipulated value can be limited to a selected value using the LMNLIMIT function. Signaling bits indicate when a limit is exceeded by the input variable.

The LMN\_NORM function normalizes the output of LMNLIMIT according to the following formula:

$$LMN = (\text{output of LMNLIMIT}) * LMN\_FAC + LMN\_OFF$$

LMN\_FAC has the default 1 and LMN\_OFF the default 0.

The manipulated value is also available in the peripheral format. The CPR\_OUT function converts the floating-point value LMN to a peripheral value according to the following formula:

$$LMN\_PER = LMN * \frac{27648}{100}$$

**Feedforward Control**

A disturbance variable can be fed forward at the **DISV** input.

**Modes****Complete Restart/Restart**

SFB41 "CONT\_C" has a complete restart routine that is run through when the input parameter COM\_RST = TRUE is set.

During startup, the integrator is set internally to the initialization value I\_ITVAL. When it is called in a cyclic interrupt priority class, it then continues to work starting at this value.

All other outputs are set to their default values.

**Error Information**

The error output parameter RET\_VAL is not used.

## Block Diagram

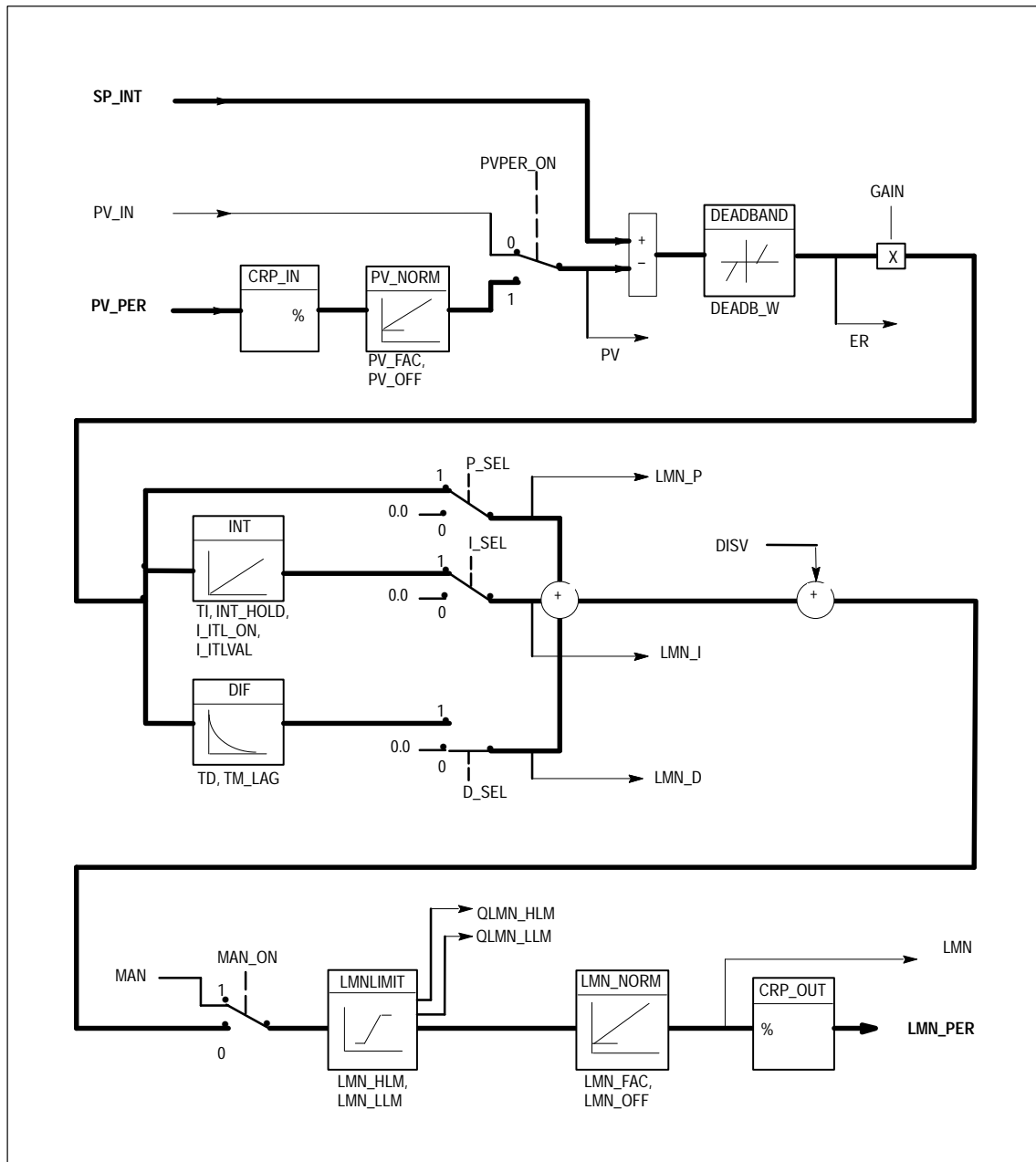


Figure 22-1 Block Diagram of CONT\_C

**Input Parameters**

Table 22-1 contains the description of the input parameters for SFB41 “CONT\_C”.

Table 22-1 Input Parameters (INPUT) for SFB41 “CONT\_C”

Parameter	Data Type	Range of Values	Default	Description
COM_RST	BOOL		FALSE	COMPLETE RESTART The block has a complete restart routine that is processed when the input “complete restart” is set.
MAN_ON	BOOL		TRUE	MANUAL VALUE ON If the input “manual value on” is set, the control loop is interrupted. A manual value is set as the manipulated value.
PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERAL ON If the process variable is read from the I/Os, the input PV_PER must be connected to the I/Os and the input “process variable peripheral on” must be set.
P_SEL	BOOL		TRUE	PROPORTIONAL ACTION ON The PID actions can be activated or deactivated individually in the PID algorithm. The P action is on when the input “proportional action on” is set.
I_SEL	BOOL		TRUE	INTEGRAL ACTION ON The PID actions can be activated or deactivated individually in the PID algorithm. The I action is on when the input “integral action on” is set.
INT_HOLD	BOOL		FALSE	INTEGRAL ACTION HOLD The output of the integrator can be “frozen” by setting the input “integral action hold”.
I_ITL_ON	BOOL		FALSE	INITIALIZATION OF THE INTEGRAL ACTION ON The output of the integrator can be connected to the input I_ITL_VAL by setting the input “initialization of the integral action on”.
D_SEL	BOOL		FALSE	DERIVATIVE ACTION ON The PID actions can be activated or deactivated individually in the PID algorithm. The D action is on when the input “derivative action on” is set.
CYCLE	TIME	$\geq 1\text{ms}$	T#1s	SAMPLING TIME The time between the block calls must be constant. The “sampling time” input specifies the time between block calls.
SP_INT	REAL	-100.0 to +100.0 (%) or phys. value 1)	0.0	INTERNAL SETPOINT The “internal setpoint” input is used to specify a setpoint.

Table 22-1 Input Parameters (INPUT) for SFB41 "CONT\_C", continued

Parameter	Data Type	Range of Values	Default	Description
PV_IN	REAL	–100.0 to +100.0 (%) or phys. value 1)	0.0	PROCESS VARIABLE IN An initialization value can be set at the "process variable in" input or an external process variable in floating point format can be connected.
PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERAL The process variable in the I/O format is connected to the controller at the "process variable peripheral" input.
MAN	REAL	–100.0 to +100.0 (%) or phys. value 2)	0.0	MANUAL VALUE The "manual value" input is used to set a manual value using the operator interface functions.
GAIN	REAL		2.0	PROPORTIONAL GAIN The "proportional value" input specifies the controller gain.
TI	TIME	>= CYCLE	T#20s	RESET TIME The "reset time" input determines the time response of the integrator.
TD	TIME	>= CYCLE	T#10s	DERIVATIVE TIME The "derivative time" input determines the time response of the derivative unit.
TM_LAG	TIME	>= CYCLE/2	T#2s	TIME LAG OF THE DERIVATIVE ACTION The algorithm of the D action includes a time lag that can be assigned at the "time lag of the derivate action" input.
DEADB_W	REAL	>= 0.0 (%) or phys. value 1)	0.0	DEAD BAND WIDTH A dead band is applied to the error. The "dead band width" input determines the size of the dead band.
LMN_HLM	REAL	LMN_LLM ...100.0 (%) or phys. value 2)	100.0	MANIPULATED VALUE HIGH LIMIT The manipulated value is always limited by an upper and lower limit. The "manipulated value high limit" input specifies the upper limit.
LMN_LLM	REAL	–100.0... LMN_HLM (%) or phys. value 2)	0.0	MANIPULATED VALUE LOW LIMIT The manipulated value is always limited by an upper and lower limit. The "manipulated value low limit" input specifies the lower limit.
PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the process variable. The input is used to adapt the process variable range.
PV_OFF	REAL		0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the process variable. The input is used to adapt the process variable range.



Table 22-1 Input Parameters (INPUT) for SFB41 “CONT\_C”, continued

Parameter	Data Type	Range of Values	Default	Description
LMN_FAC	REAL		1.0	MANIPULATED VALUE FACTOR The “manipulated value factor” input is multiplied by the manipulated value. The input is used to adapt the manipulated value range.
LMN_OFF	REAL		0.0	MANIPULATED VALUE OFFSET The “manipulated value offset” is added to the manipulated value. The input is used to adapt the manipulated value range.
I_ITLVAL	REAL	–100.0 to +100.0 (%) or phys. value 2)	0.0	INITIALIZATION VALUE OF THE INTEGRAL ACTION The output of the integrator can be set at input I_ITL_ON. The initialization value is applied to the input “initialization value of the integral action”.
DISV	REAL	–100.0 to +100.0 (%) or phys. value 2)	0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to input “disturbance variable”.

1) Parameters in the setpoint and process variable branches with the same unit

2) Parameters in the manipulated value branch with the same unit

## Output Parameters

Table 22-2 contains the description of the output parameters for SFB41 “CONT\_C”.

Table 22-2 Output Parameters (OUTPUT) for SFB41 “CONT\_C”

Parameter	Data Type	Range of Values	Default	Description
LMN	REAL		0.0	MANIPULATED VALUE The effective manipulated value is output in floating point format at the “manipulated value” output.
LMN_PER	WORD		W#16#0000	MANIPULATED VALUE PERIPHERAL The manipulated value in the I/O format is connected to the controller at the “manipulated value peripheral” output.
QLMN_HLM	BOOL		FALSE	HIGH LIMIT OF MANIPULATED VALUE REACHED The manipulated value is always limited to an upper and lower limit. The output “high limit of manipulated value reached” indicates that the upper limit has been exceeded.
QLMN_LLM	BOOL		FALSE	LOW LIMIT OF MANIPULATED VALUE REACHED The manipulated value is always limited to an upper and lower limit. The output “low limit of manipulated value reached” indicates that the lower limit has been exceeded.

Table 22-2 Output Parameters (OUTPUT) for SFB41 “CONT\_C”, continued

Parameter	Data Type	Range of Values	Default	Description
LMN_P	REAL		0.0	PROPORTIONAL COMPONENT The “proportional component” output contains the proportional component of the manipulated variable.
LMN_I	REAL		0.0	INTEGRAL COMPONENT The “integral component” output contains the integral component of the manipulated value.
LMN_D	REAL		0.0	DERIVATIVE COMPONENT The “derivative component” output contains the derivative component of the manipulated value.
PV	REAL		0.0	PROCESS VARIABLE The effective process variable is output at the “process variable” output.
ER	REAL		0.0	ERROR SIGNAL The effective error is output at the “error signal” output.

## 22.2 Step Control with SFB42 “CONT\_S”

**Introduction** SFB42 “CONT\_S” is used on SIMATIC S7 programmable logic controllers to control technical processes with digital manipulated value output signals for integrating actuators. During parameter assignment, you can activate or deactivate subfunctions of the PI step controller to adapt the controller to the process.

**Application** You can use the controller as a PI fixed setpoint controller or in secondary control loops in cascade, blending or ratio controllers, however not as the primary controller. The functions of the controller are based on the PI control algorithm of the sampling controller supplemented by the functions for generating the binary output signal from the analog actuating signal.

**Description** Apart from the functions in the process value branch, the SFB implements a complete PI controller with a digital manipulated value output and the option of influencing the manipulated value manually. The step controller operates without a position feedback signal.

In the following you will find the description of the partial functions:

### Setpoint Branch

The setpoint is entered in floating-point format at the **SP\_INT** input.

### Process Variable Branch

The process variable can be input in the peripheral (I/O) or floating-point format. The **CRP\_IN** function converts the **PV\_PER** peripheral value to a floating-point format of -100 to +100 % according to the following formula:

$$\text{Output of CPR\_IN} = \text{PV\_PER} * \frac{100}{27648}$$

The **PV\_NORM** function normalizes the output of **CRP\_IN** according to the following formula:

$$\text{Output of PV\_NORM} = (\text{output of CPR\_IN}) * \text{PV\_FAC} + \text{PV\_OFF}$$

**PV\_FAC** has a default of 1 and **PV\_OFF** a default of 0.

### Error Signal

The difference between the setpoint and process variable is the error signal. To suppress a small constant oscillation due to the manipulated variable quantization (for example due to a limited resolution of the manipulated value by the actuator valve), a dead band is applied to the error signal (**DEADBAND**). If **DEADB\_W** = 0, the dead band is switched off.

### PI Step Algorithm

The SFB operates without a position feedback signal. The I action of the PI algorithm and the assumed position feedback signal are calculated in **one** integrator (INT) and compared with the remaining P action as a feedback value. The difference is applied to a three-step element (THREE\_ST) and a pulse generator (PULSEOUT) that creates the pulses for the actuator. The switching frequency of the controller can be reduced by adapting the threshold on of the three-step element.

### Feedforward Control

A disturbance variable can be fed forward at the **DISV** input.

### Modes

#### Complete Restart/Restart

SFB42 “CONT\_S” has a complete restart routine that is run through when the input parameter COM\_RST = TRUE is set.

All other outputs are set to their default values.

### Error Information

The error output parameter RET\_VAL is not used.

## Block Diagram

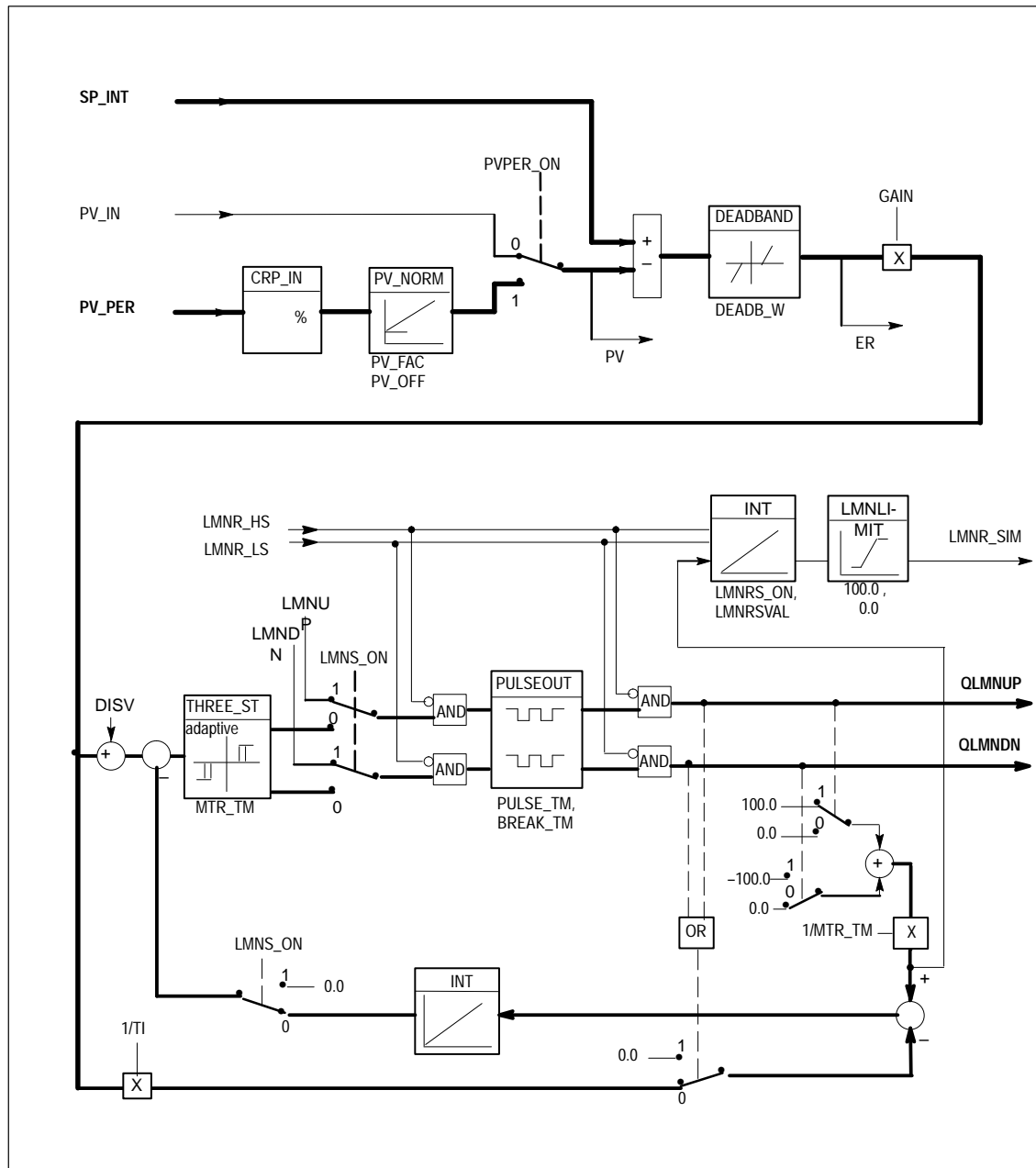


Figure 22-2 Block Diagram of CONT\_S

**Input  
Parameters**

Table 22-3 contains the description of the input parameters for SFB42 “CONT\_S”.

Table 22-3 Input Parameters (INPUT) for SFB42 “CONT\_S”

Parameter	Data Type	Values	Default	Description
COM_RST	BOOL		FALSE	COMPLETE RESTART The block has a complete restart routine that is processed when the input “complete restart” is set.
LMNR_HS	BOOL		FALSE	HIGH LIMIT OF POSITION FEEDBACK SIGNAL The “actuator at upper limit stop” signal is connected to the “high limit of position feedback signal” input. LMNR_HS=TRUE means the actuator is at upper limit stop.
LMNR_LS	BOOL		FALSE	LOW LIMIT OF POSITION FEEDBACK SIGNAL The “actuator at lower limit stop” signal is connected to the “low limit of position feedback signal” input. LMNR_LS=TRUE means the actuator is at lower limit stop.
LMNS_ON	BOOL		TRUE	MANUAL ACTUATING SIGNALS ON The actuating signal processing is switched to manual at the “manual actuating signals on” input.
LMNUP	BOOL		FALSE	ACTUATING SIGNALS UP With manual actuating value signals, the output signal QLMNUP is set at the input “actuating signals up”.
LMNDN	BOOL		FALSE	ACTUATING SIGNALS DOWN With manual actuating value signals, the output signal QLMNDN is set at the input “actuating signals down”.
PVPER_ON	BOOL		FALSE	PROCESS VARIABLE PERIPHERAL ON If the process variable is read in from the I/Os, the input PV_PER must be connected to the I/Os and the input “process variable peripheral on” must be set.
CYCLE	TIME	$\geq 1\text{ms}$	T#1s	SAMPLING TIME The time between the block calls must be constant. The “sampling time” input specifies the time between block calls.
SP_INT	REAL	-100.0 ... +100.0 (%) or phys. value 1)	0.0	INTERNAL SETPOINT The “internal setpoint” input is used to specify a setpoint.
PV_IN	REAL	-100.0 ... +100.0 (%) or phys. value 1)	0.0	PROCESS VARIABLE IN An initialization value can be set at the “process variable in” input or an external process variable in floating point format can be connected.
PV_PER	WORD		W#16#0000	PROCESS VARIABLE PERIPHERAL The process variable in the I/O format is connected to the controller at the “process variable peripheral” input.

Table 22-3 Input Parameters (INPUT) for SFB42 "CONT\_S", continued

Parameter	Data Type	Values	Default	Description
GAIN	REAL		2.0	PROPORTIONAL GAIN The "proportional gain" input sets the controller gain.
TI	TIME	>= CYCLE	T#20s	RESET TIME The "reset time" input determines the time response of the integrator.
DEADB_W	REAL	0.0...100.0 (%) or phys. value 1)	1.0	DEAD BAND WIDTH A dead band is applied to the error. The "dead band width" input determines the size of the dead band.
PV_FAC	REAL		1.0	PROCESS VARIABLE FACTOR The "process variable factor" input is multiplied by the process variable. The input is used to adapt the process variable range.
PV_OFF	REAL		0.0	PROCESS VARIABLE OFFSET The "process variable offset" input is added to the process variable. The input is used to adapt the process variable range.
PULSE_TM	TIME	>= CYCLE	T#3s	MINIMUM PULSE TIME A minimum pulse duration can be assigned with the parameter "minimum pulse time".
BREAK_TM	TIME	>= CYCLE	T#3s	MINIMUM BREAK TIME A minimum break duration can be assigned with the parameter "minimum break time".
MTR_TM	TIME	>= CYCLE	T#30s	MOTOR ACTUATING TIME The time required by the actuator to move from limit stop to limit stop is entered at the "motor actuating time" parameter.
DISV	REAL	-100.0...100.0 (%) or phys. value 2)	0.0	DISTURBANCE VARIABLE For feedforward control, the disturbance variable is connected to input "disturbance variable".

1) Parameters in the setpoint and process variable branches with the same unit

2) Parameters in the manipulated value branch with the same unit

**Output  
Parameters**

Table 22-4 contains the description of the output parameters for SFB42 “CONT\_S”.

Table 22-4 Output Parameters (OUTPUT) for SFB42 “CONT\_S”

Parameter	Data Type	Values	Default	Description
QLMNUP	BOOL		FALSE	ACTUATING SIGNAL UP If the output “actuating signal up” is set, the actuating valve is opened.
QLMNDN	BOOL		FALSE	ACTUATING SIGNAL DOWN If the output “actuating signal down” is set, the actuating valve is opened.
PV	REAL		0.0	PROCESS VARIABLE The effective process variable is output at the “process variable” output.
ER	REAL		0.0	ERROR SIGNAL The effective error is output at the “error signal” output.



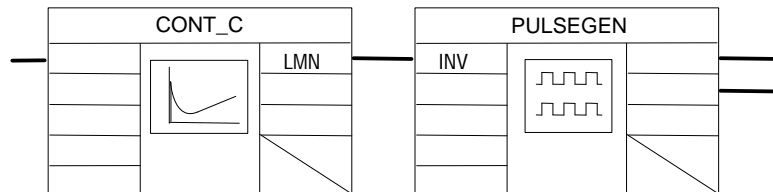
## 22.3 Pulse Generation with SFB43 “PULSEGEN”

### Introduction

SFB43 “PULSEGEN” is used to structure a PID controller with pulse output for proportional actuators

### Application

Using SFB43 “PULSEGEN”, PID two or three step controllers with pulse duration modulation can be configured. The function is normally used in conjunction with the continuous controller ~CONT\_C”.



### Description

The PULSEGEN function transforms the input variable INV (= manipulated value of the PID controller) by modulating the pulse duration into a pulse train with a constant period, corresponding to the cycle time at which the input variable is updated and which must be assigned in PER\_TM.

The duration of a pulse per period is proportional to the input variable. The cycle assigned to PER\_TM is not identical to the processing cycle of the SFB “PULSEGEN”. The PER\_TM cycle is made up of several processing cycles of SFB “PULSEGEN”, whereby the number of SFB “PULSEGEN” calls per PER\_TM cycle is the yardstick for the accuracy of the pulse duration modulation.

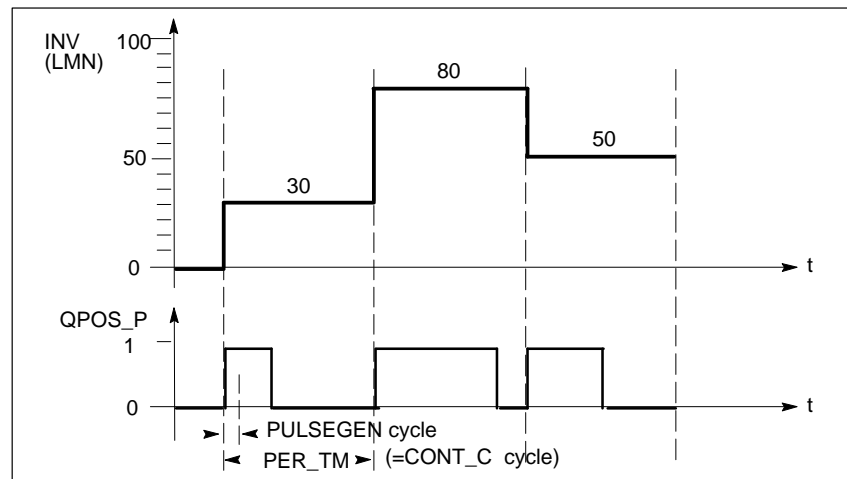


Figure 22-3 Pulse Duration Modulation

An input variable of 30% and 10 SFB “PULSEGEN” calls per PER\_TM means the following:

- “One” at the QPOS output for the first three calls of SFB “PULSEGEN” (30% of 10 calls)
- “Zero” at the QPOS output for seven further calls of SFB “PULSEGEN” (70% of 10 calls)

### Block Diagram

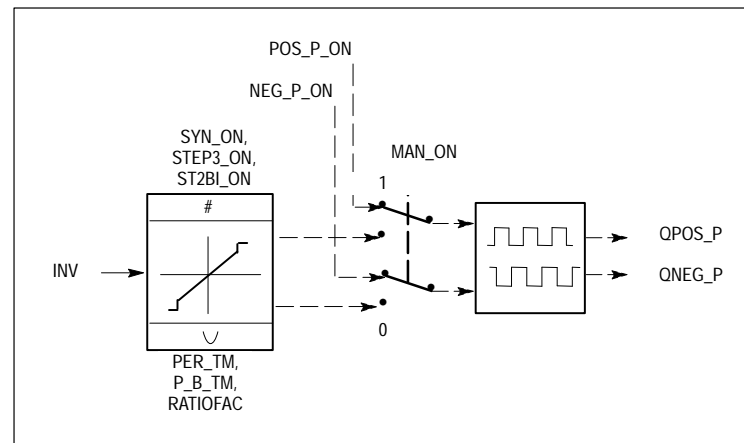


Figure 22-4 Block Diagram

### Accuracy of the Manipulated Value

With a “sampling ratio” of 1:10 (CONT\_C calls to PULSEGEN calls) the accuracy of the manipulated value in this example is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS output in steps of 10 %.

The accuracy is increased as the number of SFB “PULSEGEN” calls per CONT\_C call is increased.

If PULSEGEN is called, for example 100 times more often than CONT\_C, a resolution of 1 % of the manipulated value range is achieved.

### Note

The call frequency must be programmed by the user.

### Automatic Synchronization

It is possible to synchronize the pulse output with the block that updates the input variable INV (for example CONT\_C). This ensures that a change in the input variable is output as quickly as possible as a pulse.

The pulse generator evaluates the input value INV at intervals corresponding to the period PER\_TM and converts the value into a pulse signal of corresponding length.

Since, however, INV is usually calculated in a slower cyclic interrupt class, the pulse generator should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV.

To allow this, the block can synchronize the start of the period using the following procedure:

If INV changes and if the block call is not in the first or last two call cycles of a period, the synchronization is performed. The pulse duration is recalculated and in the next cycle is output with a new period (see Figure 22-5).

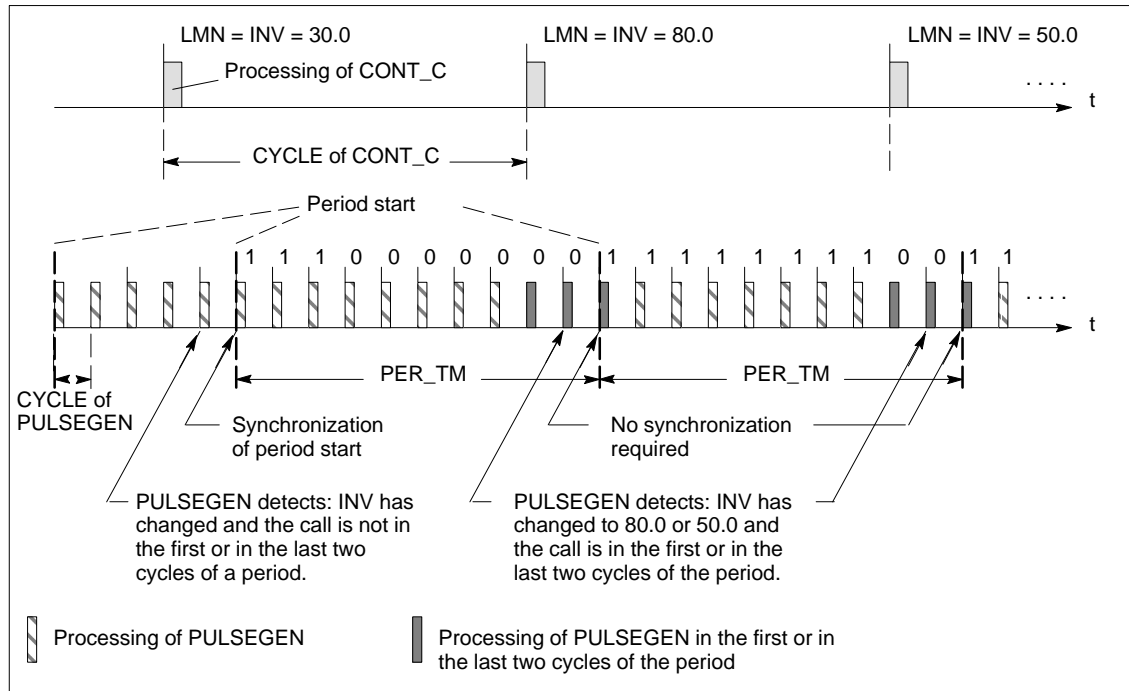


Figure 22-5 Synchronization of the Period Start

The automatic synchronization can be disabled at the “SYN\_ON” input (= FALSE).

#### Note

With the beginning of a new period, the old value of INV (in other words, of LMN) is simulated in the pulse signal more or less accurately following the synchronization.

**Modes**

Depending on the parameters assigned to the pulse generator, PID controllers with a three-step output or with a bipolar or monopolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

<b>Mode \ Switch</b>	<b>MAN_ON</b>	<b>STEP3_ON</b>	<b>ST2BI_ON</b>
Three-step control	FALSE	TRUE	Any
Two-step control with bipolar control range (-100 % to +100 %)	FALSE	FALSE	TRUE
Two-step control with monopolar control range (0 % ... 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

### Three-Step Control

In the “three-step control” mode, the actuating signal can adopt three states. The values of the binary output signals QPOS\_P and QNEG\_P are assigned to the statuses of the actuator.

The table shows the example of a temperature control:

Actuator \ Output signal	Heat	Off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Based on the input variable, a characteristic curve is used to calculate a pulse duration. The form of the characteristic curve is defined by the minimum pulse or minimum break time and the ratio factor (see Figure 22-6). The normal value for the ratio factor is 1. The “doglegs” in the curves are caused by the minimum pulse or minimum break times.

#### Minimum Pulse or Minimum Break Time

A correctly assigned minimum pulse or minimum break time P\_B\_TM can prevent short on/off times that reduce the working life of switching elements and actuators.

#### Note

Small absolute values at the input variable LMN that could otherwise generate a pulse duration shorter than P\_B\_TM are suppressed. Large input values that would generate a pulse duration longer than (PER\_TM – P\_B\_TM) are set to 100 % or -100 %.

The duration of the positive or negative pulses is calculated from the input variable (in %) multiplied by the period time.

$$\text{Period Time} = \frac{\text{INV}}{100} * \text{PER\_TM}$$

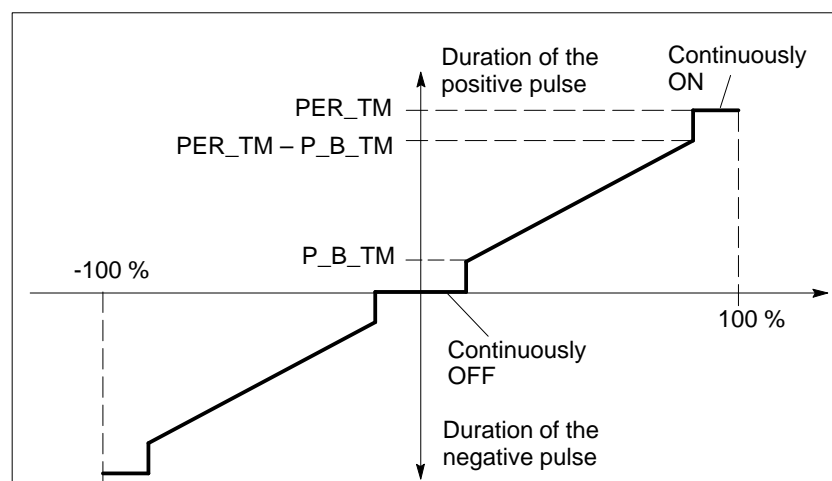


Figure 22-6 Symmetrical Characteristic Curve of the Three-Step Controller (Ratio Factor = 1)

### Three-Step Control Asymmetrical

Using the ratio factor **RATIOFAC**, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling. The ratio factor also influences the minimum pulse or minimum break time. A ratio factor  $< 1$  means that the threshold value for negative pulses is multiplied by the ratio factor.

#### Ratio Factor $< 1$

The pulse duration at the negative pulse output calculated from the input variable multiplied by the period time is reduced by the ratio factor (see Figure 22-7).

$$\text{Duration of the positive pulse} = \frac{\text{INV}}{100} * \text{PER\_TM}$$

$$\text{Duration of the negative pulse} = \frac{\text{INV}}{100} * \text{PER\_TM} * \text{RATIOFAC}$$

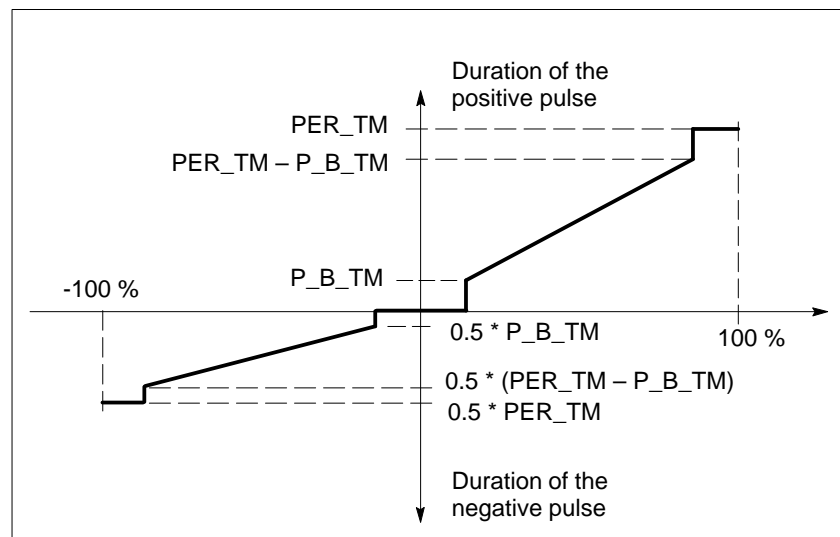


Figure 22-7 Asymmetrical Characteristic Curve of the Three-Step Controller (Ratio Factor = 0.5)

#### Ratio Factor $> 1$

The pulse duration at the positive pulse output calculated from the input variable multiplied by the period time is reduced by the ratio factor.

$$\text{Duration of the negative pulse} = \frac{\text{INV}}{100} * \text{PER\_TM}$$

$$\text{Duration of the positive pulse} = \frac{\text{INV}}{100} * \frac{\text{PER\_TM}}{\text{RATIOFAC}}$$

## Two-Step Control

In two-step control, only the positive pulse output QPOS\_P of PULSEGEN is connected to the on/off actuator. Depending on the manipulated value range being used, the two-step controller has a bipolar or a monopolar manipulated value range (see Figures 22-8 and 22-9).

### Two-Step Control with Bipolar Manipulated Variable Range (-100% to 100%)

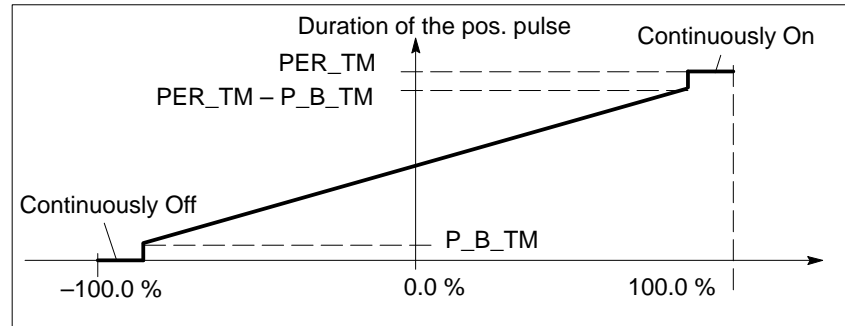


Figure 22-8 Characteristic Curve with Bipolar Manipulated Value Range (-100 % to 100 %)

### Two-Step Control with Unipolar Manipulated Variable Range (0% to 100%)

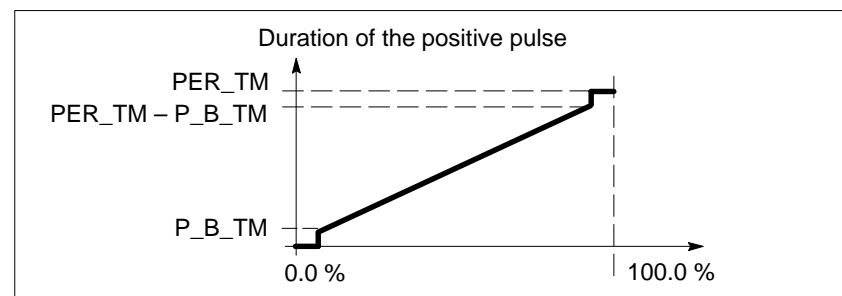


Figure 22-9 Characteristic Curve with Monopolar Manipulated Value Range (0 % to 100 %)

The negated output signal is available at QNEG\_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse \ Actuator	On	Off
	QPOS_P	QNEG_P
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

**Manual Mode in Two/Three-Step Control**

In the manual mode (MAN\_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS\_P\_ON and NEG\_P\_ON regardless of INV.

	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

**Modes****Complete Restart/Restart**

During a complete restart, all the signal outputs are set to 0.

**Error Information**

The error output parameter RET\_VAL is not used.



## Input Parameters

Table 22-5 Input Parameters (INPUT) for SFB43 “PULSEGEN”

Parameter	Data Type	Range of Values	Default	Description
INV	REAL	–100.0...100.0 (%)	0.0	<b>INPUT VARIABLE</b> An analog manipulated value is connected to the input parameter “input variable”.
PER_TM	TIME	$\geq 20 \cdot \text{CYCLE}$	T#1s	<b>PERIOD TIME</b> The constant period of pulse duration modulation is input with the “period time” input parameter. This corresponds to the sampling time of the controller. The ratio between the sampling time of the pulse generator and the sampling time of the controller determines the accuracy of the pulse duration modulation.
P_B_TM	TIME	$\geq \text{CYCLE}$	T#0ms	<b>MINIMUM PULSE/BREAK TIME</b> A minimum pulse or minimum break time can be assigned at the input parameters “minimum pulse or minimum break time”.
RATIOFAC	REAL	0.1 ...10.0	1.0	<b>RATIO FACTOR</b> The input parameter “ratio factor” can be used to change the ratio of the duration of negative to positive pulses. In a thermal process, this would, for example, allow different time constants for heating and cooling to be compensated (for example, in a process with electrical heating and water cooling).
STEP3_ON	BOOL		TRUE	<b>THREE STEP CONTROL ON</b> The “three-step control on” input parameter activates this mode. In three-step control, both output signals are active.
ST2BI_ON	BOOL		FALSE	<b>TWO STEP CONTROL FOR BIPOLAR MANIPULATED VALUE RANGE ON</b> With the input parameter “two-step control for bipolar manipulated value range on” you can select between the modes “two-step control for bipolar manipulated value” and “two-step control for monopolar manipulated value range”. The parameter STEP3_ON = FALSE must be set.
MAN_ON	BOOL		FALSE	<b>MANUAL MODE ON</b> By setting the input parameter “manual mode on”, the output signals can be set manually.
POS_P_ON	BOOL		FALSE	<b>POSITIVE PULSE ON</b> In the manual mode with three-step control, the output signal QPOS_P can be set at the input parameter “positive pulse on”. In the manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
NEG_P_ON	BOOL		FALSE	<b>NEGATIVE PULSE ON</b> In the manual mode with three-step control, the output signal QNEG_P can be set at the input parameter “negative pulse on”. In the manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.

Table 22-5 Input Parameters (INPUT) for SFB43 “PULSEGEN”, continued

Parameter	Data Type	Range of Values	Default	Description
SYN_ON	BOOL		TRUE	<b>SYNCHRONIZATION ON</b> By setting the input parameter “synchronization on”, it is possible to synchronize automatically with the block that updates the input variable INV. This ensures that a changing input variable is output as quickly as possible as a pulse.
COM_RST	BOOL		FALSE	<b>COMPLETE RESTART</b> The block has a complete restart routine that is processed when the “complete restart” input is set.
CYCLE	TIME	$\geq 1\text{ms}$	T#10ms	<b>SAMPLING TIME</b> The time between block calls must be constant. The “sampling time” input specifies the time between block calls.

**Note**

The values of the input parameters are not limited in the block. There is no parameter check.

**Output Parameters**

Table 22-6 Output Parameters (OUTPUT) for SFB43 “PULSEGEN”

Parameter	Data Type	Values	Default	Description
QPOS_P	BOOL		FALSE	<b>OUTPUT POSITIVE PULSE</b> The output parameter “output positive pulse” is set when a pulse is to be output. In three-step control, this is always the positive pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.
QNEG_P	BOOL		FALSE	<b>OUTPUT NEGATIVE PULSE</b> The output parameter “output negative pulse” is set when a pulse is to be output. In three-step control, this is always the negative pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.

## 22.4 Example of the PULSEGEN Block

### Control Loop

With the continuous controller CONT\_C and the pulse generator PULSEGEN, you can implement a fixed setpoint controller with a switching output for proportional actuators. Figure 22-10 shows the signal flow of the control loop.

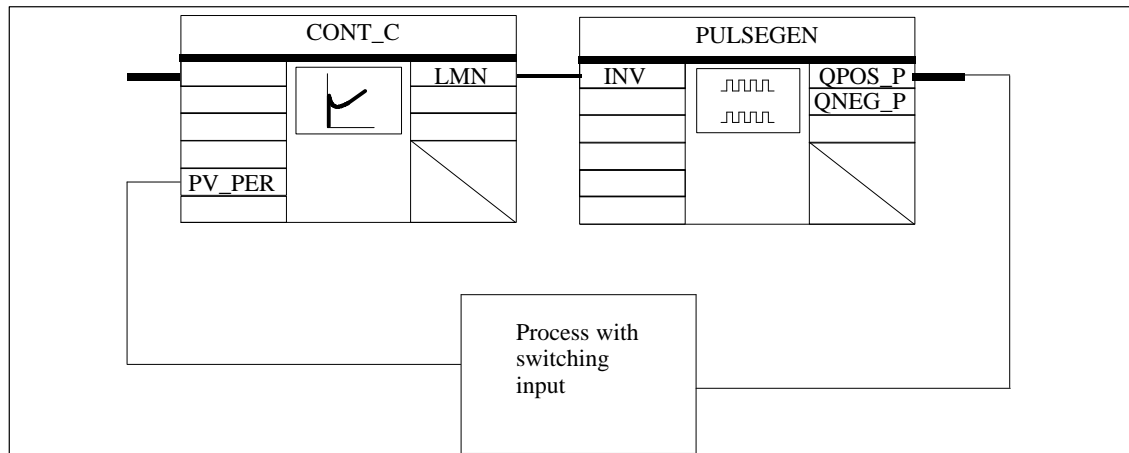


Figure 22-10 Control Loop

The continuous controller CONT\_C forms the manipulated value LMN that is converted by the pulse generator PULSEGEN into pulse/break signals QPOS\_P or QNEG\_P.

### Calling the Block and Connecting It

The fixed setpoint controller with switching output for proportional actuators PULS\_CTR consists of the blocks CONT\_C and PULSEGEN. The block call is implemented so that CONT\_C is called every 2 seconds ( $=\text{CYCLE} \times \text{RED\_FAC}$ ) and PULSEGEN every 10 ms ( $=\text{CYCLE}$ ). The cycle time of OB35 is set to 10 ms. The interconnection can be seen in Figure 22-11.

During a complete restart, the block PULS\_CTR is called in OB100 and the input COM\_RST is set to TRUE.

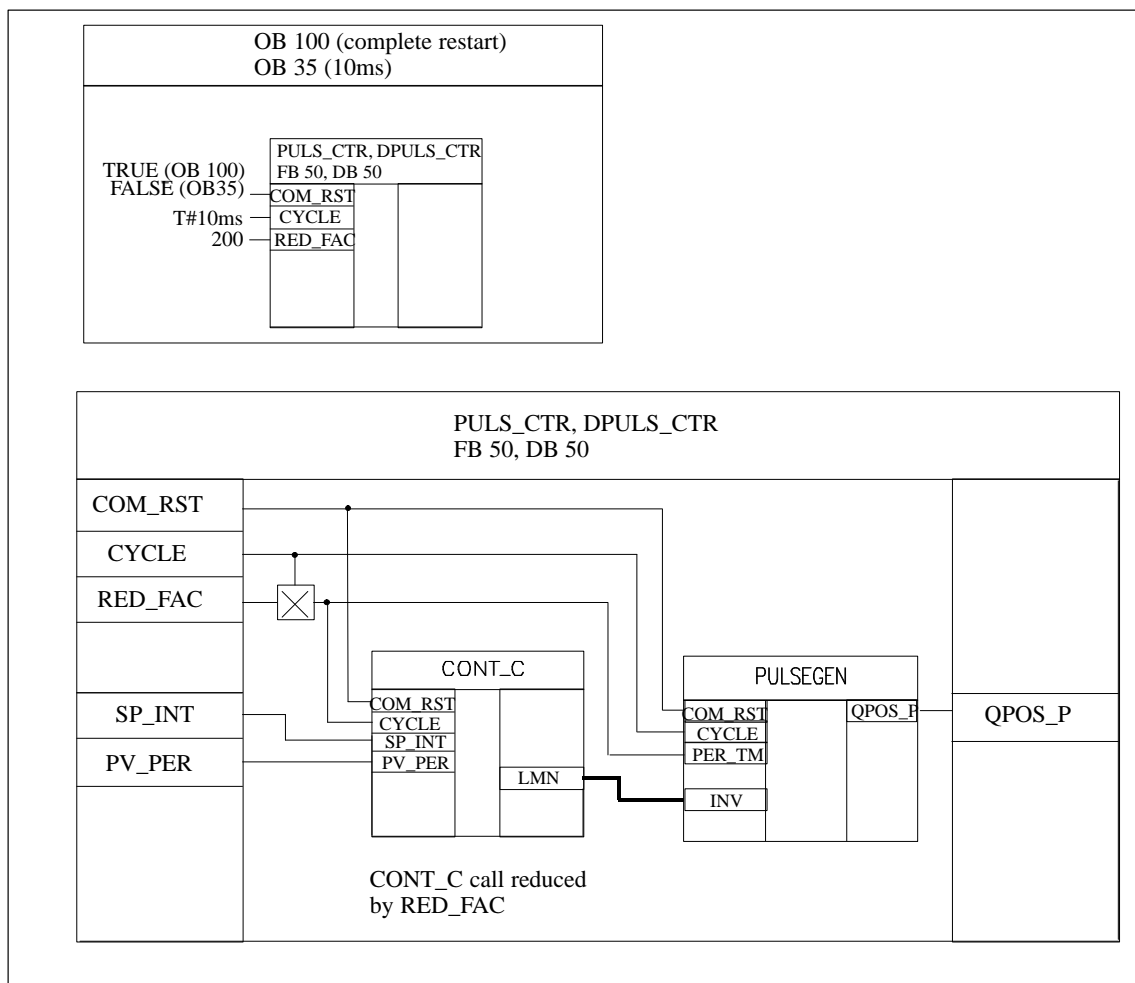


Figure 22-11 Block Call and Interconnection

### STL Program for FB PULS\_CTR

Table 22-7 FB PULS\_CTR

Address	Declaration	Name	Type	Comment
0.0	in	SP_INT	REAL	Setpoint
4.0	in	PV_PER	WORD	Process variable peripheral
6.0	in	RED_FAC	INT	Call reduction factor
8.0	in	COM_RST	BOOL	Complete restart
10.0	in	CYCLE	TIME	Sampling time
14.0	out	QPOS_P	BOOL	Actuating signal
16.0	stat	DI_CONT_C	FB-CONT_C	Counter
142.0	stat	DI_PULSEGEN	FB-PULSEGEN	Counter
176.0	stat	sCount	INT	Counter
0.0	temp	tCycCtr	TIME	Controller sampling time

Table 22-8 Network 1

STL	Description
A       #COM_RST	//Complete restart routine
JCN     M001	
L       0	
T       #sCount	
M001: L   #CYCLE	//Calculate controller sampling time
L       #RED_FAC	
*D	
T       #tCycCtr	
L       #sCount	//Decrement counter and compare with zero
L       1	
-I	
T       #sCount	
L       0	
<=I	
JCN     M002	//Conditional block call and set counter
CALL    #DI_CONT_C	
COM_RST :=#COM_RST	
CYCLE   :=#tCycCtr	
SP_INT   :=#SP_INT	
PV_PER   :=#PV_PER	
L       #RED_FAC	
T       #sCount	
M002: L   #DI_CONT_C.LMN	
T       #DI_PULSEGEN.INV	
CALL    #DI_PULSEGEN	
PER_TM   :=#tCycCtr	
COM_RST :=#COM_RST	
CYCLE   :=#CYCLE	
QPOS_P   :=#QPOS_P	
BE	



# Diagnostic Data

# A

Section	Description	Page
A.1	Overview of the Structure of Diagnostic Data	A-2
A.2	Diagnostic Data	A-3
A.3	Structure of Channel-Specific Diagnostic Data	A-5

## A.1 Overview of the Structure of Diagnostic Data

### Data Record 0 and 1 of the System Data

The diagnostic data of a module are located in data records 0 and 1 of the system data area (see Section 7.1).

- Data record 0 contains 4 bytes of diagnostic data that describe the current status of a signal module.
- Data record 1 contains
  - the 4 bytes of diagnostic data, also located in data record 0 **and**
  - the diagnostic data specific to the module.

### Structure and Content of the Diagnostic Data

This section describes the structure and content of the individual bytes of the diagnostic data.

Whenever an error occurs, the corresponding bit is set to “1”.



## A.2 Diagnostic Data

Table A-1 Structure and Content of the Diagnostic Data

Byte	Bit	Meaning	Remarks	
0	0	Module fault		
	1	Internal error		
	2	External error		
	3	Channel error		
	4	No external auxiliary voltage		
	5	No front connector		
	6	No parameter assignment		
	7	Wrong parameters in the module		
1	0 to 3	Module class	0101 0000 1000 1100 1111	Analog module CPU Function module CP Digital module
	4	Channel information exists		
	5	User information exists		
	6	Diagnostic interrupt from substitute		
	7	0		
2	0	No or wrong memory card		
	1	Communication problem		
	2	Mode	0 1	RUN STOP
	3	Cycle monitoring responded		
	4	Internal module supply voltage failed		
	5	Battery exhausted		
	6	Entire battery backup failed		
	7	0		
3	0	Expansion rack failure		
	1	Processor failure		
	2	EPROM error		
	3	RAM error		
	4	ADC/DAC error		
	5	Fuse tripped		
	6	Hardware interrupt lost		
	7	0		

Table A-1 Structure and Content of the Diagnostic Data, continued

Byte	Bit	Meaning	Remarks	
4	0 to 6	Channel type	B#16#70 B#16#72 B#16#71 B#16#73 B#16#74 B#16#75 B#16#76 B#16#77 B#16#78 B#16#79 to B#16#7D B#16#7E B#16#7F	Digital input Digital output Analog input Analog output FM-POS FM-REG FM-ZAEHL FM-TECHNO FM-NCU reserved US300 reserved
	7	Further channel type exists?	0 1	no yes
5	0 to 7	Number of diagnostic bits output per channel by a module.	The number of diagnostic bits per channel is rounded up to byte boundaries.	
6	0 to 7	Number of channels of one channel type on a module.	If different channel types exist on a module, the structure is repeated in data record 1 from byte 4 onwards for each channel type.	
7	0	Channel error channel 0/ Channel group 0	First byte of the channel error vector (the length of the channel error vector depends on the number of channels and is rounded up to a byte boundary).	
	1	Channel error channel 1/ Channel group 1		
	2	Channel error channel 2/ Channel group 2		
	3	Channel error channel 3/ Channel group 3		
	4	Channel error channel 4/ Channel group 4		
	5	Channel error channel 5/ Channel group 5		
	6	Channel error channel 6/ Channel group 6		
	7	Channel error channel 7/ Channel group 7		
...	—	Channel-specific errors (see Section A.3)		

### A.3 Structure of Channel-Specific Diagnostic Data

#### Channel-Specific Errors

Starting at the byte immediately following the channel error vector, the channel-specific errors are indicated for each channel of the module. The tables below show the structure of channel-specific diagnostic data for the different channel types. The bits have the following meaning:

- 1 = Error
- 0 = No error

#### Analog Input Channel

Table A-2 Diagnostic Byte for an Analog Input Channel

Bit	Meaning	Remarks
0	Configuration/ parameter assignment error	Can be signaled by SFC52 and EVENTN = W#16#8x50
1	Common mode error	Can be signaled by SFC52 and EVENTN = W#16#8x51
2	P short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x52
3	M short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x53
4	Wire break/supply current monitoring Measuring transducer/Pt 100	Can be signaled by SFC52 and EVENTN = W#16#8x54
5	Reference channel error	Can be signaled by SFC52 and EVENTN = W#16#8x55
6	Current below measuring range (< 3 mA)	Can be signaled by SFC52 and EVENTN = W#16#8x56
7	Current above measuring range (> 22 mA)	Can be signaled by SFC52 and EVENTN = W#16#8x57

**Analog Output Channel**

Table A-3 Diagnostic Byte for an Analog Output Channel

Bit	Meaning	Remarks
0	Configuration/ parameter assignment error	Can be signaled by SFC52 and EVENTN = W#16#8x60
1	Common mode error	Can be signaled by SFC52 and EVENTN = W#16#8x61
2	P short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x62
3	M short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x63
4	Wire break/supply current monitoring Measuring transducer/Pt 100	Can be signaled by SFC52 and EVENTN = W#16#8x64
5	0	reserved
6	No load voltage	Can be signaled by SFC52 and EVENTN = W#16#8x66
7	0	reserved

**Digital Input Channel**

Table A-4 Diagnostic Byte for a Digital Input Channel

Bit	Meaning	Remarks
0	Configuration/parameter assignment error	Can be signaled by SFC52 and EVENTN = W#16#8x70
1	Ground error	Can be signaled by SFC52 and EVENTN = W#16#8x71
2	P short circuit (sensor)	Can be signaled by SFC52 and EVENTN = W#16#8x72
3	M short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x73
4	Wire break	Can be signaled by SFC52 and EVENTN = W#16#8x74
5	No sensor power supply	Can be signaled by SFC52 and EVENTN = W#16#8x75
6	0	reserved
7	0	reserved

**Digital Output Channel**

Table A-5 Diagnostic Byte for a Digital Output Channel

Bit	Meaning	Remarks
0	Configuration/parameter assignment error	Can be signaled by SFC52 and EVENTN = W#16#8x80
1	Ground error	Can be signaled by SFC52 and EVENTN = W#16#8x81
2	P short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x82
3	M short circuit	Can be signaled by SFC52 and EVENTN = W#16#8x83
4	Wire break	Can be signaled by SFC52 and EVENTN = W#16#8x84
5	Fuse tripped	Can be signaled by SFC52 and EVENTN = W#16#8x86
6	No load voltage	Can be signaled by SFC52 and EVENTN = W#16#8x86
7	Overtemperature	Can be signaled by SFC52 and EVENTN = W#16#8x87



# System Status Lists (SZL)

## Chapter Overview

Section	Description	Page
B.1	Overview of the System Status Lists (SZL)	B-2
B.2	Structure of an SZL Partial List	B-3
B.3	SZL-ID	B-4
B.4	Possible Partial System Status Lists	B-5
B.5	SZL-ID W#16#xy00 - List of Available SZL-IDs of a Module	B-6
B.6	SZL-ID W#16#xy11 – Module Identification	B-7
B.7	SZL-ID W#16#xy12 – CPU Characteristics	B-8
B.8	SZL-ID W#16#xy13 – User Memory Areas	B-10
B.9	SZL-ID W#16#xy14 – System Areas	B-12
B.10	SZL-ID W#16#xy15 – Block Types	B-13
B.11	SZL-ID W#16#xy16 – Existing Priority Classes	B-14
B.12	SZL-ID W#16#xy17 – List of Permitted SDBs	B-15
B.13	SZL-ID W#16#xy18 – Maximum S7-300 I/O Configuration	B-16
B.14	SZL-ID W#16#xy21 – Status of Module LEDs	B-17
B.15	SZL-ID W#16#xy21 – Assignment of Interrupts and Errors	B-18
B.16	SZL-ID W#16#xy22 – Interrupt Status	B-20
B.17	SZL-ID W#16#xy23 – Status of the Priority Classes	B-22
B.18	SZL-ID W#16#xy24 – Operating Mode and Mode Transition	B-24
B.19	SZL-ID W#16#xy31 – Capability Parameters for Communication	B-28
	Sections B.20 to B.30 contain the data records of all extracts of partial list W#16#0131	
B.31	SZL-ID W#16#xy32 – Communication Status Data	B-42
	Sections B.32 to B.43 contain the data records of all extracts of partial list W#16#0132	
B.44	SZL-ID W#16#xy33 – Stations for S7 Messages and Diagnostic Events	B-55
B.45	SZL-ID W#16#xy81 – Local Data of the OBs	B-56
B.46	SZL-ID W#16#xy82 – Start Events	B-57
B.47	SZL-ID W#16#xy91 – Module Status Information	B-58
B.48	SZL-ID W#16#xy92 – Rack / Station Status Information	B-61
B.49	SZL-ID W#16#xyA0 – Diagnostic Buffer	B-64
B.50	SZL-ID W#16#00B1 – Module Diagnostic Information	B-65
B.51	SZL-ID W#16#00B2 – Module Diagnostic Data with Geographical Address	B-66
B.52	SZL-ID W#16#00B3 – Module Diagnostic Data with Logical Base Address	B-67
B.53	SZL-ID W#16#00B4 Diagnostic Data of a DP Slave	B-68

## B.1 Overview of the System Status Lists (SZL)

### Overview of This Appendix

This appendix describes all the partial lists of the system status list that relate to the following:

- CPUs
- Modules whose partial lists are not module-specific (for example SZL-IDs W#16#00B1, W#16#00B2, W#16#00B3).

Module-specific partial lists, for example, for CPs and FMs are included in the descriptions of the particular modules.

### Definition: System Status List

The system status list (SZL) describes the current status of a programmable logic controller. The content of the SZL can only be read using information functions but cannot be modified. The partial lists are virtual lists, in other words, they are only created by the operating system of the CPUs when specifically requested.

You can only read one system status list using SFC 51 “RDSYSST”.

### Content

The system status lists contain information about the following:

- System data
- Diagnostic status data in the CPU
- Diagnostic data on modules
- Diagnostic buffer

### System Data

System data are fixed or assigned characteristic data of a CPU. They provide information about the following:

- The configuration of the CPU
- The status of the priority classes
- Communication

### Diagnostic Status Data

Diagnostic status data describe the current status of the components monitored by system diagnostic functions.

### Diagnostic Data on Modules

The modules with diagnostic capabilities assigned to a CPU have diagnostic data that are stored directly on the module.

### Diagnostic Buffer

The diagnostic buffer contains diagnostic entries in the order in which they occur.



## B.2 Structure of a Partial SZL List

### Basics

You can read partial lists and partial list extracts

- Using information functions on the programming device
- With SFC 51 “RDSYSST”.

If you use SFC51 “RDSYSST”, you specify what you want to read using the parameters SZL\_ID and INDEX.

### Structure

A partial list consists of the following:

- A header and
- The data records.

### Header

The header of a partial list consists of the following:

- SZL-ID
- Index
- Length of a data record of the partial list in bytes
- Number of data records contained in the partial list.

### Index

With certain partial lists or partial list extracts an object type ID or an object number must be specified. The index is used for this purpose. If it is not required for the information, its content is irrelevant.

### Data Records

A data record in a partial list has a specific length. This depends on the information in the partial list. How the data words in a data record are used also depends on the particular partial list.

## B.3 SZL-ID

### SZL-ID

Every partial system status list has a number. You can output a complete partial list or an extract from it. The possible partial list extracts are predefined and are identified by a number. The SZL-ID consists of the number of the partial list, the number of the partial list extract and the module class.

### Structure

The SZL-ID is one word long. The meaning of the bits in the SZL-ID is as follows:

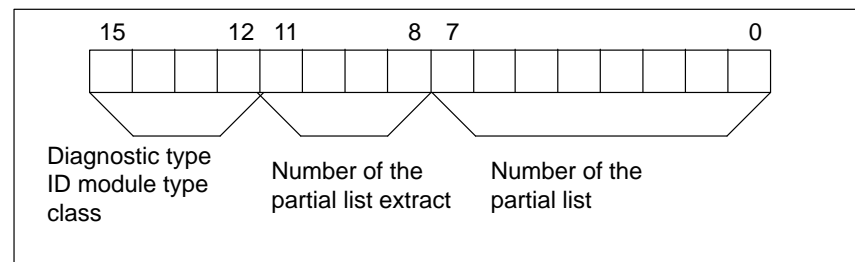


Figure B-1 Structure of the SZL-ID

### Module Class

Examples of module classes:

Table B-1 Module Type Classes

Module Class	Coding (binary)
CPU	0000
CP	1100
FM	1000

### Number of the Partial List Extract

The number of the partial list extracts and their meaning depend on the particular system status list to which they belong. With the number of the partial list extract, you specify which subset of a partial list you want to read.

### Number of the Partial List

Using the number of the partial list, you specify which partial list of the system status list you want to read.

## B.4 Possible Partial System Status Lists

### Subset

Any one module only has a subset of all the possible partial lists. Which partial lists are available depends on the particular module.

### Possible SZL Partial Lists

The following table lists all the possible partial lists with the number contained in the SZL-ID.

Partial List	SZL-ID
List of all the SZL-IDs of a module	W#16#xy00
Module identification	W#16#xy11
CPU characteristics	W#16#xy12
User memory areas	W#16#xy13
System areas	W#16#xy14
Block types	W#16#xy15
Priority classes	W#16#xy16
List of the permitted SDBs with a number < 1000	W#16#xy17
Maximum S7-300 I/O configuration	W#16#xy18
Status of the module LEDs	W#16#xy19
Interrupt / error assignment	W#16#xy21
Interrupt status	W#16#xy22
Priority classes	W#16#xy23
Modes	W#16#xy24
Communication capability parameters	W#16#xy31
Communication status data	W#16#xy32
Diagnostics: device logon list	W#16#xy33
Start information list	W#16#xy81
Start event list	W#16#xy82
Module status information	W#16#xy91
Rack / station status information	W#16#xy92
Diagnostic buffer of the CPU	W#16#xyA0
Module diagnostic information (data record 0)	W#16#00B1
Module diagnostic information (data record 1), geographical address	W#16#00B2
Module diagnostic information (data record 1), logical address	W#16#00B3
Diagnostic data of a DP slave	W#16#00B4

## B.5 SZL-ID W#16#xy00 - List of Available SZL-IDs of a Module

**Purpose** If you read the partial lists with SZL-ID W#16#xy00, you obtain the SZL-IDs supported by the module.

**Header** The header of system status list SZL-ID W#16#xy00 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16# 0000H: all SZL partial lists of the module W#16# 0100H: a partial list with all partial list extracts W#16#0200H: a partial list extract W#16#0300H: possible indexes of a partial list extract W#16#0F00H: only partial list header information
INDEX	see below
LENGTHDR	W#16#0002: one data record is 1 word long (2 bytes)
N_DR	Number of data records

**Index** The INDEX parameter has the following meaning depending on the particular partial list extract:

SZL-ID	INDEX	Meaning
0000	0000	All SZL-IDs of a module
0100	Z0YY	A partial list with all partial list extracts Z: module type class YY: number of the SZL partial list
0200	ZXYY	A partial list extract Z: module type class X: number of the partial list extract YY: number of the SZL partial list
0300	ZXYY	Possible indexes of an SZL partial list Z: module type class X: number of the partial list extract YY: number of the SZL partial list
0F00	0000	Partial list header information (number of all the SZL-IDs of the module)

**Data Record** A data record of partial list SZL-ID W#16#xy00 has the following structure:

Name	Length	Meaning
SZL-ID	1 word	SZL-ID/index, that exists If you selected SZL_ID = W#16#0300 <ul style="list-style-type: none"> <li>0 means that no index is permitted or the index is irrelevant</li> <li>W#16#FFFF means that the indexes of this partial list cannot be ascertained</li> </ul>

## B.6 SZL-ID W#16#xy11 – Module Identification

**Purpose** If you read the system status list with SZL-ID W#16#xy11, you obtain the module identification of the module.

**Header** The header of system status list SZL-ID W#16#xy11 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0011: all identification data records of a module W#16#0111: a single identification data record W#16#0F11: only partial list header information
INDEX	Only with SZL-ID W#16#0111: Number of a particular data record W#16#0001: identification of the module
LENGTHDR	W#16#001C: one data record is 14 words long (28 bytes)
N_DR	Number of data records

**Data Record** A data record of system status list SZL-ID W#16#xy11 has the following structure:

Name	Length	Meaning
Index	1 word	Index of an identification data record
MlfB	10 words	Order number of the module: string of 19 characters and a blank (20H); for example for CPU 314: "6ES7 314-0AE01-0AB0 "
BGTyp	1 word	Module type ID (the module type ID is only used for internal purposes)
Ausbg	1 word	Version of the module or release of the operating system
Ausbe	1 word	Release of the PG description file

## B.7 SZL-ID W#16#xy12 – CPU Characteristics

**Purpose** CPU modules have different characteristics depending on the hardware being used. Each characteristic is assigned an ID. If you read the partial list with SZL-ID W#16#xy12, you obtain the characteristics of the module.

**Header** The header of partial list SZL-ID W#16#xy12 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract: W#16#0012: all characteristics W#16#0112: characteristics of a group You specify the group in the INDEX parameter. W#16#0F12: partial list header information
INDEX	Group W#16#0000: MC5 processing unit W#16#0100: time system W#16#0200: system response W#16#0300: language description of the CPU
LENGTHDR	W#16#0002: one data record is 1 word long (2 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy12 is one word long. An identifier is entered for each characteristic. A characteristics identifier is one word long.

**Characteristics Identifier** The following table lists all the characteristics identifiers.

Identifier	Meaning
W#16#0000 – 00FF	MC7 processing unit (group with index 0000)
W#16#0001	MC7 processing generating code
W#16#0002	MC7 interpreter
W#16#0100 – 01FF	Time system (group with index 0100)
W#16#0101	1 ms resolution
W#16#0102	10 ms resolution

Identifier	Meaning
W#16#0103	No real time clock
W#16#0104	BCD time-of-day format
W#16#0200 – 02FF	System response (group with index 0200)
W#16#0201	Capable of multiprocessor mode
W#16#0300 – 03FF	MC7 Language description of the CPU (group with index 0300)
W#16#0301	Reserved
W#16#0302	All 32 bit fixed-point instructions
W#16#0303	All floating-point instructions
W#16#0304	sin,asin,cos,acos,tan,atan,sqr,sqrt,ln,exp
W#16#0305	Accumulator 3/accumulator 4 with corresponding instructions (ENT,PUSH,POP,LEAVE)
W#16#0306	Master Control Relay instructions
W#16#0307	Address register 1 exists with corresponding instructions
W#16#0308	Address register 2 exists with corresponding instructions
W#16#0309	Operations for area-crossing addressing
W#16#030A	Operations for area-internal addressing
W#16#030B	All memory-indirect addressing instructions for bit memory (M)
W#16#030C	All memory-indirect addressing instructions for data blocks (DB)
W#16#030D	All memory-indirect addressing instructions for data blocks (DI)
W#16#030E	All memory-indirect addressing instructions for local data (L)
W#16#030F	All instructions for parameter transfer in FCs
W#16#0310	Memory bit edge instructions for process image input (I)
W#16#0311	Memory bit edge instructions for process image output (Q)
W#16#0312	Memory bit edge instructions for bit memory (M)
W#16#0313	Memory bit edge instructions for data blocks (DB)
W#16#0314	Memory bit edge instructions for data blocks (DI)
W#16#0315	Memory bit edge instructions for local data (L)
W#16#0316	Dynamic evaluation of the FC bit
W#16#0317	Dynamic local data area with the corresponding instructions
W#16#0318	Reserved
W#16#0319	Reserved

## B.8 SZL-ID W#16#xy13 – Memory Areas

**Purpose** If you read the partial list with SZL-ID W#16#xy13, you obtain information about the memory areas of the module.

**Header** The header of partial list SZL-ID W#16#xy13 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0013: data records of all memory areas W#16#0113: data record for one memory area You specify the memory area with the INDEX parameter. W#16#0F13: only partial list header information
INDEX	Specifies a memory area (only with SZL-ID W#16#0113) W#16#0001: work memory W#16#0002: load memory integrated W#16#0003: load memory plugged in W#16#0004: maximum plug-in load memory W#16#0005: size of the backup memory W#16#0006: size of the memory reserved by the system for CFBs
LENGTHDR	W#16#0024: one data record is 18 words long (36 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy13 has the following structure:

Name	Length	Meaning
Index	1 word	Index of a memory area W#16#0001: work memory W#16#0002: load memory integrated W#16#0003: load memory plugged in W#16#0004: maximum plug-in load memory W#16#0005: size of the backup memory W#16#0006: size of the memory reserved by the system for CFBs
Code	1 word	Memory type: W#16#0001: volatile memory (RAM) W#16#0002: non-volatile memory (FEPR0M) W#16#0003: mixed memory (RAM + FEPR0M)
Size	2 words	Total size of the selected memory (total of area 1 and area 2)



Name	Length	Meaning
Mode	1 word	Logical mode of the memory Bit 0: volatile memory area Bit 1: non-volatile memory area Bit 2: mixed memory area For work memory: Bit 3: code and data separate Bit 4: code and data together
Granu	1 word	Always has the value 0
Ber1	2 words	Size of the volatile memory area in bytes If the INDEX is W#16#0006: work memory occupied by CFBs
Belegt1	2 words	Size of the volatile memory area being used
Block1	2 words	Largest free block in the volatile memory area If 0: no information available or cannot be ascertained.
Ber2	2 words	Size of the non-volatile memory area in bytes If the INDEX is W#16#0006: system memory occupied by CFBs
Belegt2	2 words	Size of the non-volatile memory area being used
Block2	2 words	Largest free block in the non-volatile memory area If 0: no information available or cannot be ascertained.

## B.9 SZL-ID W#16#xy14 – System Areas

**Purpose** If you read the partial list with SZL-ID W#16#xy14, you obtain information about the system areas of the module.

**Header** The header of partial list SZL-ID W#16#xy14 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0014: all system areas of a module W#16#0114: one system area, specified with the INDEX parameter W#16#0F14: only for partial list header information
INDEX	Only for SZL-ID W#16#0114: W#16#0001: PII (number in bytes) W#16#0002: PIQ (number in bytes) W#16#0003: memory (number) W#16#0004: Timers (number) W#16#0005: counters (number) W#16#0006: number of bytes in the logical address area W#16#0007: size of the entire local data area of the CPU in bytes
LENGTHDR	W#16#0008: one data record is 4 words long (8 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy14 has the following structure:

Name	Length	Meaning
Index	1 word	Index of the system area W#16#0001: PII (number in bytes) W#16#0002: PIQ (number in bytes) W#16#0003: memory (number) W#16#0004: timers (number) W#16#0005: counters (number) W#16#0006: number of bytes in the logical address area W#16#0007: local data (entire local data area of the module in bytes)
code	1 word	Memory type W#16#0001: volatile memory (RAM) W#16#0002: non-volatile memory (FEPR0M) W#16#0003: mixed memory (RAM and FEPR0M)
anzahl	1 word	Number of elements of the system area
reman	1 word	Number of retentive elements

## B.10 SZL-ID W#16#xy15 – Block Types

**Purpose** If you read the partial list with SZL-ID W#16#xy15, you obtain the block types that exist on the module.

**Header** The header of partial list SZL-ID W#16#xy15 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0015: Data records of all block types of a module W#16#0115: Data record of a block type You specify the block type using the INDEX parameter. W#16#0F15: Only partial list header information
INDEX	Only for SZL-ID W#16#0115: W#16#0800: OB W#16#0A00: DB W#16#0B00: SDB W#16#0C00: FC W#16#0E00: FB
LENGTHDR	W#16#0006: one data record is 5 words long (10 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy15 has the following structure:

Name	Length	Meaning
Index	1 word	Block type number W#16#0800: OB W#16#0A00: DB W#16#0B00: SDB W#16#0C00: FC W#16#0E00: FB
MaxAnz	1 word	Maximum number of blocks of the type OBs: max. possible number of OBs for a CPU DBs: max. possible number of DBs including DB0 SDBs: max. possible number of SDBs including SDB2 FCs and FBs: max. possible number of loadable blocks
MaxLng	1 word	Maximum total size of the object to be loaded in Kbytes
Maxabl	2 words	Maximum length of the work memory part of a block in bytes

## B.11 SZL-ID W#16#xy16 – Existing Priority Classes

**Purpose** If you read the partial list with SZL-ID W#16#xy16, you obtain information about the priority classes that exist on the module.

**Header** The header of partial list SZL-ID W#16#xy16 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0016: data records of all priority classes W#16#0116: data record of the specified priority class You specify the priority class using the INDEX parameter. W#16#0F16: only partial list header info
INDEX	Index of a priority class W#16#0000: free cycle W#16#000A: time-of-day interrupt W#16#0014: time-delay interrupt W#16#001E: cyclic interrupt W#16#0028: hardware interrupt W#16#0050: asynchronous error interrupt W#16#005A: background W#16#0064: startup W#16#0078: Synchronous error interrupt
LENGTHDR	W#16#0006: one data record is 3 words long (6 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy16 has the following structure:

Name	Length	Meaning
Index	1 word	See the table above
maxanz	1 word	Maximum number of OBs in the priority class
anzakt	1 word	Number of chained OBs in the priority class

## B.12 SZL-ID W#16#xy17 – List of Permitted SDBs

**Purpose** If you read the partial list with SZL-ID W#16#xy17, you obtain the list of permitted system data blocks (SDBs) of the module. You can only list SDBs with a number less than 1000.

**Header** The header of partial list SZL-ID W#16#xy17 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0017: all SDBs of a module W#16#0117: one single SDB You specify the SDB number using the INDEX parameter. W#16#0F17: only partial list header information
INDEX	Only with SZL-ID W#16#0117: Number of the SDB
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy17 has the following structure:

Name	Length	Meaning
SDBNr	1 word	Number of the SDB
state	1 word	Characteristics of the SDB Bit 0: 0: SDB cannot be copied / cannot be linked 1: SDB can be copied / can be linked Bit 1: 0: SDB not generated as default 1: SDB generated as default Bit 2 to 15: Reserved

**SDB Number** If you have selected SZL-ID 0117, and you specify an illegal SDB as the SDB number, the parameter RET\_VAL of SFC51 contains the error code W#16#8083 (index wrong).

### B.13 SZL-ID W#16#xy18 – Maximum S7-300 I/O Configuration

#### Purpose

If you read the partial system status list with SZL-ID W#16#??18, you obtain the maximum I/O configuration for modules of the S7-300. You obtain the possible numbers of the racks and the number of slots.

You can also check the maximum possible number of racks and the total number of all possible slots.

#### Header

The header of partial list W#16#xy18 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0018: all data records W#16#0118: one data record You specify the number of the rack using the INDEX parameter. W#16#0F18: Only partial list header information
INDEX	Only with SZL-ID = W#16#0118: 0, 1, 2, 3: Number of the rack W#16#00FF: Maximum number of racks (racknr) and total number of possible slots (anzst)
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	Number of data records

#### Data Record

A data record of partial list SZL-ID W#16#xy18 has the following structure:

Name	Length	Meaning
racknr	1 word	Number of the rack 0: Rack 0 1: Rack 1 2: Rack 2 3: Rack 3 If you specify the index W#16#00FF before the SZL-ID 0118H: maximum number of racks
anzst	1 word	Maximum number of slots per rack If you have specified the index W#16#00FF for SZL-ID 0118H, this means the total number of possible slots

## B.14 SZL-ID W#16#xy19 – Status of the Module LEDs

**Purpose** If you read the partial list with SZL-ID W#16#xy19, you obtain the status of the module LEDs.

**Header** The header of partial list W#16#xy19 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial system status list W#16#0019 Status of all LEDs W#16#0119 Status of one LED. You select the LED using the INDEX parameter. W#16#0F19 Only partial list header information
INDEX	LED identifier (only relevant for SZL-ID W#16#0119) 1: SF (group error) 2: INTF (internal error) 3: EXTf (external error) 4: RUN 5: STOP 6: FRCE (force) 7: CRST (complete restart) 8: BAF (battery problem/overload, battery voltage shorted on bus) 9: USR (user-defined)
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	Number of data records

**Data Record** A data record of the partial list with SZL-ID W#16#xy19 has the following structure:

Name	Length	Meaning
index	1 word	LED identifier
led_on	1 byte	Status of the LED: 0: off 1: on
led_blink	1 byte	Flashing status of the LED: 0: not flashing 1: flashing normally (2 Hz) 2: flashing slowly (0.5 Hz)

## B.15 SZL-ID W#16#xy21 – Assignment of Interrupts and Errors

### Purpose

If you read the partial list with SZL-ID W#16#xy21, you obtain information about how interrupts and errors are assigned to OBs.

### Header

The header of partial list SZL-ID W#16#xy21 is structured as follows:

Content	Meaning
SZL-ID	<p>The SZL-ID of the partial system status list</p> <p>W#16#0021 Data records of all possible interrupts on a module</p> <p>W#16#0121 Data records of all possible interrupts of one class</p> <p>You specify the interrupt class using the INDEX parameter.</p> <p>W#16#0221 Data record for the specified interrupt.</p> <p>You specify the interrupt (OB no.) using the INDEX parameter.</p> <p>W#16#0921 Data records of all interrupts of one class and for which the corresponding interrupt OB is loaded.</p> <p>You specify the interrupt class using INDEX.</p> <p>W#16#0A21 Data records of all interrupts for which the corresponding interrupt OB is loaded</p> <p>W#16#0F21 Only partial list header information</p>
INDEX	<p>Interrupt/error class/OB number (with SZL-ID W#16#0221)</p> <p>W#16#0000: free cycle</p> <p>W#16#0A0A: time-of-day interrupt</p> <p>W#16#1414: time-delay interrupt</p> <p>W#16#1E23: cyclic interrupt</p> <p>W#16#2828: hardware interrupt</p> <p>W#16#5050: asynchronous errors interrupts</p> <p>W#16#005A: background</p> <p>W#16#0064: startup</p> <p>W#16#7878: synchronous errors interrupts</p>
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	Number of data records



**Data Record**

A data record of the partial list with SZL-ID W#16#xy21 has the following structure:

Name	Length	Meaning
ereig	1 word	Event ID of the start event If an OB has several start events, the event with the lowest defined number is displayed in the form W#16#x0zz; x: event class, zz: event number. The event ID is explained in <b>/70/</b> and <b>/101/</b> .
ae	1 byte	Number of the assigned priority class If the priority class cannot be specified, ae has the following meaning: B#16#00 OB was deselected with STEP 7 B#16#FE OB not loaded or disabled or discarded by SFC B#16#7F Synchronous error OB. Masking of synchronous error OBs is not displayed. If several causes occur at the same time, the cause with the lowest number is displayed.
ob	1 byte	OB number

## B.16 SZL-ID W#16#xy22 – Interrupt Status

### Purpose

If you read the partial list with SZL-ID W#16#xy22, you obtain information about the current status of interrupt processing and the interrupts generated by the module.

### Header

The header of partial list SZL-ID W#16#xy22 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial system status list W#16#0022 Data records of all possible interrupts on a module W#16#0122 Data records of all possible interrupts of a class. You specify the interrupt class using the INDEX parameter. W#16#0222 Data record for the specified interrupt You specify the interrupt (OB no.) using the INDEX parameter. W#16#0822 Data records of all interrupts of a class for which the corresponding interrupt OB is loaded You specify the interrupt class using the INDEX parameter. W#16#0922 Data records of all interrupts for which the OB is loaded W#16#0F22 Only partial list header
INDEX	Priority class or OB no. (with SZL-ID W#16#0222) W#16#0000: free cycle W#16#0A0A: time-of-day interrupt W#16#1414: time-delay interrupt W#16#1E23: cyclic interrupt W#16#2828: hardware interrupt W#16#5050: asynchronous error interrupt W#16#005A: background W#16#0064: startup W#16#7878: synchronous error interrupt
LENGTHDR	W#16#001C: one data record is 14 words long (28 bytes)
N_DR	Number of data records

**Data Record**

A data record of partial list SZL-ID W#16#xy22 has the following structure:

Name	Length	Meaning
info	10 words	<p>Start information for the corresponding OB, with the following exceptions:</p> <ul style="list-style-type: none"> <li>• For OB1, the current minimum and maximum cycle times are available.</li> <li>• For OB80, the configured minimum and maximum cycle times can be read.</li> <li>• For error interrupts without the current information</li> <li>• For interrupts, the status information contains the current parameter assignment of the source of the interrupt.</li> <li>• For synchronous errors, B#16#7F is entered as the priority class if the OBs have not yet been executed, otherwise the priority class of the last call.</li> </ul> <p>If an OB has several start events and if these have not occurred when the information is fetched, event number W#16#xyzz is returned with x: event class, zz: lowest defined number of group, y: undefined. Otherwise the number of the last start event to occur is used.</p>
al 1	1 word	<p>Processing identifier</p> <p>Bit 0: Interrupt event disabled/enabled by parameter assignment = 0: enabled = 1: disabled</p> <p>Bit 1: Interrupt event disabled by SFC 39 "DIS_IRT" = 0: not disabled = 1: disabled</p> <p>Bit 2 = 1: Interrupt source is active (generate job exists for time interrupts, time-of-day interrupt OB started, time-delay interrupt OB started, cyclic interrupt OB: time started)</p> <p>Bit 4: Interrupt OB = 0: not loaded = 1: loaded</p> <p>Bit 5: Interrupt OB disabled by test and installation function = 1: disabled</p>
al 2	1 word	<p>Reaction if OB not loaded/disabled</p> <p>Bit 0 = 1: Disable interrupt source</p> <p>Bit 1 = 1: Generate interrupt event error</p> <p>Bit 2 = 1: CPU changes to the STOP mode</p> <p>Bit 3 = 1: Only discard interrupt</p>
al 3	2 words	<p>Discard by test and installation functions:</p> <p>Bit no. x set means: the event number that is x higher than the lowest event number of the corresponding OB is discarded by the test and installation function.</p>

## B.17 SZL-ID W#16#xy23 – Status of the Priority Classes

### Purpose

If you read the partial list SZL-ID W#16#xy23, you obtain information about the priority classes of the module.

### Header

The header of partial list SZL-ID W#16#xy23 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0023 data records of all priority classes of a module W#16#0123 data record of one priority class You specify the priority class with the INDEX parameter. W#16#0223 data records of the priority classes being processed W#16#0F23 only partial list header information
INDEX	Priority class The following values are possible W#16#0000 to W#16#001C
LENGTHDR	W#16#0012: one data record is 9 words long (18 bytes)
N_DR	Number of data records

**Data Record**

A data record of partial list SZL-ID W#16#xy23 has the following structure:

Name	Length	Meaning
ae	1 byte	Priority class
aestat	1 byte	B#16#00: No OB being processed B#16#FE: Priority class disabled by SDB0 with STEP 7 (configuration) B#16#xx: Number of the 1st OB being processed
aefstat	1 word	Error status of the priority class Bit 0: – Bit 1: the 1st error OB is OB121 Bit 2: the 1st error OB is OB122 Bit 3: – Bit 4: – Bit 5: the 2nd error OB is OB121 Bit 6: the 2nd error OB is OB122 Bit 7: – Bit 8: – Bit 9: – Bit 10:– Bit 11 – Bit 12:Double error Bit 13:Logic error Bit 14:Stack error Bit 15:–
maxbst	1 byte	Maximum nesting of the block
maxsti	1 byte	Maximum amount of start information that can be saved
aktsiv	1 byte	Current amount of start information before processing
aktsib	1 byte	Maximum amount of start information being processed
grld	1 word	Size of the local data stack of the specified priority class
progfm	2 words	Programming error mask (see description of SFC 36 "MSK_FLT")
syncfm	2 words	Synchronous error mask (see description of SFC 36 "MSK_FLT")

## B.18 SZL-ID W#16#xy24 – Operating Mode and Mode Transition

### Purpose

If you read the system status list with SZL-ID W#16#xy24, you obtain information about the modes of the module.

### Header

The header of partial list SZL-ID W#16#xy24 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0024 all modes that can occur on the module W#16#0124 information about the last mode transition W#16#0224 processed mode transition W#16#0424 current mode transition W#16#0524: specified mode transition You specify the mode with the INDEX parameter. W#16#0F24: only partial list header information
INDEX	Only for SZL-ID W#16#0524: mode W#16#5000: mode STOP W#16#5010: mode STARTUP W#16#5020: mode RUN W#16#5030: mode HOLD W#16#4520: mode DEFECT
LENGTHDR	W#16#0014: one data record is 10 words long (20 bytes)
N_DR	Number of data records

### Data Record

A data record of partial list SZL-ID W#16#xy24 has the following structure:

Name	Length	Meaning
Info	10 words	Status information from status information

**Mode Transition Information**

The information about a mode transition (mode transition information) is 20 bytes long and is structured as follows:

Name	Length	Meaning
ereig	1 word	Event ID (see Section C.1) possible W#16#4xy?
ae	1 byte	B#16#FF
bzü-id	1 byte	ID of the mode change divided into 4 bits Bit 0 to 3: Requested mode Bit 4 to 7: Previous mode Operating mode IDs of the requested or previous modes: 1H: STOP (update) 2H: STOP (memory reset) 3H: STOP (self initialization) 4H: STOP (internal) 5H: Startup (complete restart) 7H: Restart 8H: RUN AH: HOLD DH: DEFECT
res	2 words	Reserved
anlinfo 1	1 byte	Bit 0 =0: No difference in expected and actual configuration (only S7-300) = 1: Difference in expected and actual configuration (only S7-300) Bit 1 =0: No difference in expected and actual configuration = 1: Difference in expected and actual configuration (only S7-300) Bit 3 =0: Clock for time stamp not battery backed at last POWER ON =1: Clock for time stamp battery backed at last POWER ON Bit 5 and 4. Multiprocessor mode =00B:Single processor mode =01B:Multicomputing (only S7-400) =10B:Operation of more than one CPU in a segmented rack (only S7-400)

Name	Length	Meaning
anlinfo 2	1 byte	<p>Type of startup just executed</p> <p>B#16#01: Complete restart in multicomputing without command to CPU according to parameter assignment (only S7-400)</p> <p>B#16#03: Complete restart set at mode selector</p> <p>B#16#04: Complete restart command via MPI</p> <p>B#16#0A: Restart in multicomputing without command to the CPU according to parameter assignment (only S7-400)</p> <p>B#16#0B: Restart set at mode selector (only S7-400)</p> <p>B#16#0C: Restart command via MPI (only S7-400)</p> <p>B#16#10: Automatic complete restart after battery-backed power on</p> <p>B#16#13: Complete restart set at mode selector; last power on battery backed</p> <p>B#16#14: Complete restart command via MPI; last power on battery backed</p> <p>B#16#20: Automatic complete restart after non battery backed power on (with memory reset by system)</p> <p>B#16#23: Complete restart set at mode selector; last power on unbattery backed</p> <p>B#16#24: Complete restart command via MPI; last power on unbattery backed</p> <p>B#16#A0: Automatic restart after battery backed power on according to parameter assignment (only S7-400)</p>
anlinfo 3	1 byte	<p>Permissibility of startup types</p> <p>B#16#?0: Manual startup illegal memory reset requeste)</p> <p>B#16#?1: Manual startup illegal parameter settings etc. must be changed</p> <p>B#16#?7: Manual complete restart permitted</p> <p>B#16#?F: Manual complete restart and manual restart permitted (only S7-400)</p> <p>B#16#0?: Automatic restart illegal, memory reset requested</p> <p>B#16#1?: Automatic restart illegal (parameter settings etc. must be changed)</p> <p>B#16#7?: Automatic complete restart permitted</p> <p>B#16#F?: Automatic complete restart and automatic restart permitted (only S7-400)</p>



Name	Length	Meaning
anlinfo 4	1 byte	Last valid operation or setting of the automatic startup type at power on B#16#00: No startup type B#16#01: Complete restart in multicomputing without command to CPU according to parameter assignment (only S7-400) B#16#03: Complete restart due to switch setting B#16#04: Complete restart command via MPI B#16#0A: Restart in multicomputing without command to the CPU according to parameter assignment (only S7-400) B#16#0B: Restart set at mode selector (only S7-400) B#16#0C: Restart command via MPI (only S7-400) B#16#10: Automatic complete restart after battery-backed power on B#16#13: Complete restart set at mode selector; last power on battery backed B#16#14: Complete restart command via MPI; last power on battery backed B#16#20: Automatic complete restart after non battery backed power on (with memory reset by the system) B#16#23: Complete restart set at mode selector; last power on not battery backed B#16#24: Complete restart command via MPI; last power on not battery backed B#16#A0: Automatic restart after battery backed power on according to parameter assignment (only S7-400)
time	4 words	Time stamp

### Mode Transition Information

The information about a mode transition (mode transition information) is 20 bytes long and is structured as follows:

Name	Length	Meaning
ereig	1 word	Event ID (see Section C.1) possible W#16#5???
ae	1 byte	B#16#FF
bz-id	1 byte	Operating mode identifier: B#16#01: STOP (update) B#16#02: STOP (memory reset) B#16#03: STOP (self initialization) B#16#04: STOP (internal) B#16#06: complete restart B#16#07: restart B#16#08: RUN B#16#0A: HOLD B#16#0D: DEFECT
res	4 words	Reserved
time	4 words	Time stamp

## B.19 SZL-ID W#16#xy31 – Capability Parameters for Communication

**Purpose** If you read the partial list with SZL-ID W#16#xy31, you obtain information about the communication capabilities of the module.

**Header** The header of partial list SZL-ID W#16#xy31 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0031 not defined W#16#0131 Information about a communication unit You specify the communication unit using the INDEX parameter. W#16#0F31 only partial list header information
INDEX	W#16#0001 General data for communication W#16#0002 Test and installation function constants W#16#0003 Operator interface (O/I) W#16#0004 Object management system (OMS) W#16#0005 Diagnostics W#16#0006 Communication function block (CFB) W#16#0007 Global data W#16#0008 Test and installation function time information W#16#0009 Time-of-day capability parameters W#16#0010 Message parameters W#16#0011 SCAN capability parameters
LENGTHDR	W#16#0028: one data record is 20 words long (40 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy31 is always 20 words long. The data records have different contents. The content depends on the INDEX parameter, in other words, which type of communication the data record belongs to.

## B.20 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0001

**Content** The partial list extract with SZL-ID W#16#0131 and the index W#16#0001 contains general data about the communication of a communication unit.

**Data Record** A data record of partial list extract SZL-ID W#16#0131 with index W#16#0001 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0001: Index for general communication data
pdu	1 word	Maximum PDU size in bytes
anz	1 word	Maximum number of communication connections
mpi_bps	2 words	Maximum data rate of the MPI in hexadecimal format Example: 16#2DC6C corresponds to 187500 bps
kbus_bps	2 words	Maximum data rate of the communication bus
res	13 words	Reserved

## B.21 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0002

**Content** The partial list extract with SZL-ID W#16#0131 and the index W#16#0002 contains information about the test and installation constants of the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0131 with index W#16#0002 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0002: test and installation
funkt_0 to funkt_5	6 bytes	Permitted TIS functions (bit = 1: function exists)
funkt_0	1 byte	Bit 0: reserved, bit 1: block status, bit 2: variable status, bit 3: output ISTACK, bit 4: output BSTACK, bit 5: output LSTACK, bit 6: time measurement from ... to ..., Bit 7: force selection
funkt_1	1 byte	Bit 0: modify variable, bit 1: force, bit 2: breakpoint, Bit 3: exit HOLD, bit 4: memory reset, bit 5: disable job, bit 6: enable job, bit 7: delete job
funkt_2	1 byte	Bit 0: read job list, bit 1: read job, bit 2: replace job, bit 3 to bit 7: reserved
funkt_3	1 byte	Reserved
funkt_4	1 byte	
funkt_5	1 byte	
aseg	6 bytes	Non-relevant system data
eseg	6 bytes	
trgereig_0 bis trgereig_2	3 bytes	Permitted trigger events
trgereig_0	1 byte	Bit 0: immediately Bit 1: system trigger Bit 2: system checkpoint main cycle start Bit 3: system checkpoint main cycle end Bit 4: mode transition RUN-STOP Bit 5: after code address Bit 6: code address area Bit 7: data address
trgereig_1	1 byte	Bit 0: data address area Bit 1: local data address Bit 2: local data address area Bit 3: range trigger Bit 4: before code address Bit 5 to bit 7: reserved
trgereig_2	1 byte	reserved

Name	Length	Meaning
trgbed	1 byte	System data with no relevance
pfad	1 byte	
tiefe	1 byte	
systrig	1 byte	
erg par	1 byte	
erg pat 1	1 word	
erg pat 2	1 word	
force	1 word	Number of modifiable variables
time	1 word	Upper time limit run-time meas. (Format: bits 0 to 11 contain the time value (0 to 4K-1); bits 12 to 15 contain the time base: 0H= $10^{-10}$ s, 1H = $10^{-9}$ s, ..., AH = $10^0$ s, ... FH = $10^5$ s)
res	2 words	Reserved

## B.22 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0003

### Content

The partial list extract with SZL-ID W#16#0131 and the index W#16#0003 contains information about the communication parameters of the module for connection to a unit for operator interface functions.

### Data Record

A data record of partial list extract SZL-ID W#16#0131 with index W#16#0003 has the following structure:

Name	Length	Meaning
Index	1	W#16#0003: Index for operator interface functions
funkt_0 to funkt_3	4 bytes	Available functions (to funkt_1 bit 3) and addressable areas (from funkt_1 bit 4)
funkt_0	1 byte	Bits indicating the available functions, bit 0: read once, bit 1: write once, bit 2: initialize cyclic reading (start implicitly), bit 3: initialize cyclic reading (start explicitly), bit 4: start cyclic reading, bit 5: stop cyclic reading, bit 6: clear cyclic reading, bit 7: reserved
funkt_1	1 byte	Bit 0 to bit 3: reserved, bit 4: peripheral I/Os, bit 5: inputs, bit 6: outputs, Bit 7: bit memory
funkt_2	1 byte	Bit 0: user DB, bit 1: data record, bit 2 to bit 6: reserved, bit 7: S7 counter
funkt_3	1 byte	Bit 0: S7 timer, bit 1: IEC counter, bit 2: IEC timer, bit 3: high speed counter, bit 4 to Bit 7: reserved
data	1 word	Maximum size of consistently readable data
anz	1 word	Maximum number of cyclic read jobs
per min	1 word	Minimum period for cyclic read jobs (n x 100 ms)
per max	1 word	Maximum period for cyclic read jobs (n x 100 ms)
res	13 words	Reserved

## B.23 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0004

### Content

The partial list extract with SZL-ID W#16#0131 and the index W#16#0004 contains information about the object management system (OMS) of the module.

### Data Record

A data record of partial list extract SZL-ID W#16#0131 with index W#16#0004 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0004 Index for OMS
funkt_0 to funkt_7	8 bytes	Available object management system functions: (Bit = 1: functions available on the CPU)
funkt_0	1 byte	Bit 0: reserved, bit 1: directory (hierarchy 1), bit 2: directory (hierarchy 2), bit 3: directory (hierarchy 3), bit 4: copy, bit 5: chain (list), bit 6: chain (all copied), bit 7: delete (list),
funkt_1	1 byte	Bit 0: upload on PG, bit 1: assign parameters when chaining, bit 2: LOAD function when exchanging data with CFBs,, bit 3 to bit 6: reservrd, bit 7: delete *.*
funkt_2	1 byte	Bit 0: load user program (RAM), bit 1: load user program (EPROM), bit 2: save user program (RAM), bit 3: save user program (EPROM), bit 4: save user program (all), bit 5: compress (external), bit 6: firmware update (using communication), bit 7: set RAM memory mode
funkt_3	1 byte	Bit 0: set EPROM memory mode, Bit 1 to Bit 5: reserved Bit 6: assign parameters to newly plugged in modules Bit 7: assign parameters when evaluating memory card
funkt_4	1 byte	bit 0: Assign parameters when loading user program, bit 1: assign parameters in complete restart, bit 2: assign parameters in restart, bit 3: compress (SFC25 "COMPRESS"), bit 4: evaluate memory card after switch setting, bit 5: firmware update using memory card, bit 6 to 7: reserved
funkt_5	1 byte	Reserved
funkt_6	1 byte	
funkt_7	1 byte	
kop	1 byte	Maximum number of copied blocks
del	1 byte	Maximum number of uninterruptable, deletable blocks
kett	1 byte	Maximum number of blocks chained in one job
hoch	1 byte	Maximum number of simultaneous upload procedures
ver	1 byte	Maximum size (in bytes) of shiftable blocks in RUN. With an S7-300, this size refers to the entire block, with the S7-400, it refers to the part of the block relevant to running the program.
res	25 bytes	Reserved

## B.24 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0005

**Content** The partial list extract with SZL-ID W#16#0131 and the index W#16#0005 contains information about the diagnostic capabilities of the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0131 with index W#16#0005 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0005: Diagnostics
funkt_0 to funkt_7	4	Available diagnostic functions (bit = 1 : function exists)
funkt_0	1 byte	Bit 0: reserved, bit 1: diagnostic buffer exists, bit 2: sending system diagnostic data possible, bit 3: sending user-defined diagnostic messages possible, bit 4: sending VMD status possible, bit 5: evaluating diagnostic interrupts, bit 6: diagnostic interrupt exists on module, bit 7 : reserved
funkt_1	1 byte	Reserved
funkt_2	1 byte	
funkt_3	1 byte	
funkt_4	1 byte	
funkt_5	1 byte	
funkt_6	1 byte	
funkt_7	1 byte	
anz_sen	1 word	Maximum number of diagnostic data sinks
anz_ein	1 word	Maximum number of entries in the diagnostic buffer
anz_mel	1 word	Maximum number of process control group messages
res	13 words	Reserved



## B.25 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0006

### Content

The partial list extract with SZL-ID W#16#0131 and the index W#16#0006 contains information about the functions available for data exchange with communication SFBs for configured connections on the module.

### Data Record

A data record of partial list extract SZL-ID W#16#0131 with index W#16#0006 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0006: Data exchange with communication SFBs for configured connections
funkt_0 to funkt_7	8 bytes	Block types available for data exchange with communication SFBs for configured connections
funkt_0	1 byte	Bit 0: USEND Bit 1: URCV Bit 2: SEND Bit 3: RCV Bit 4: BSEND Bit 5: BRCV Bit 6: GET Bit 7: PUT
funkt_1	1 byte	Bit 0: PRINT Bit 1: ABORT Bit 2: INITIATE Bit 3: START Bit 4: STOP Bit 5: RESUME Bit 6: STATUS Bit 7: USTATUS
funkt_2	1 byte	Bit 0: PI Bit 1: READ Bit 2: WRITE Bit 3: LOAD Bit 4: LOAD_ME Bit 5: ALARM Bit 6: ALARM_8 Bit 7: ALARM_8P
funkt_3	1 byte	Bit 0: NOTIFY Bit 1: AR_SEND Bit 2 to bit 7: reserved
funkt_4	1 byte	Reserved
funkt_5	1 byte	

Name	Length	Meaning
funkt_6	1 byte	Bit 0: X_SEND Bit 1: X_RCV Bit 2: X_GET Bit 3: X_PUT Bit 4: X_ABORT Bit 5: I_GET Bit 6: I_PUT Bit 7: I_ABORT
funkt_7	1 byte	Bit 0: SCAN_SND Bit 1: ALARM_SQ Bit 2: ALARM_S Bit 3: ALARM_SC Bit 4: EN_MSG Bit 5: DIS_MSG Bit 6: CONTROL Bit 7: Reserved
schnell	1 byte	Fast reaction yes/no
zug_typ	4 words	Permitted module types for fast reaction
zugtyp_0 bis zug- typ_7	8 byte	Permitted module types for fast reaction
zugtyp_0	1 byte	Bit assignment, see funkt_0
zugtyp_1	1 byte	Bit assignment, see funkt_1
zugtyp_2	1 byte	Bit assignment, see funkt_2
zugtyp_3	1 byte	Bit assignment, see funkt_3
zugtyp_4	1 byte	reserved
zugtyp_5	1 byte	reserved
zugtyp_6	1 byte	Bit assignment, see funkt_6
zugtyp_7	1 byte	Bit assignment, see funkt_7
res1	1 byte	Reserved
max_sd_e mpf	1 word	Maximum number of send and receive parameters per block
max_sd_al 8p	1 word	Maximum number of send parameters for ALARM_8P
max_inst	1 word	Maximum number of instances for communication SFBs for configured connections
res2	1 word	Reserved
verb_proj	1 byte	Connection configured (yes=1) possible
verb_prog	1 byte	Connection programmed (yes=1) possible
res3	10 bytes	Reserved

## B.26 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0007

### Content

The partial list extract with SZL-ID W#16#0131 and the index W#16#0007 contains information about the functions available for global data communication on the module.

### Data Record

A data record of partial list extract SZL-ID W#16#0131 with index W#16#0007 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0007: Global data communication
funkt_0 to funkt_1	2 bytes	Available GD functions
funkt_0	1 byte	Bit 0: cyclic, bit 1: GD_SND, bit 2: GD_RCV, bit 3 to bit 7: reserved
funkt_1	1 byte	Reserved
obj_0 to obj_1	2 bytes	Addressable objects (note: no P area)
obj_0	1 byte	Bit 0: M, bit 1: PII, bit 2: PIQ, bit 3: T, bit 4: C, bit 5: DB, bit 6 to bit 7: reserved
obj_1	1 byte	Reserved
kons	1 byte	Consistent length in bytes
sen	1 byte	Minimum scan rate for sending
rec	1 byte	Minimum scan rate for receiving
time	1 byte	Time monitoring when receiving yes/no
proj	1 byte	Re-configuration possible in RUN yes/no
alarm	1 byte	Communication interrupt yes/no
mode	1 byte	Party line/MPI, communication bus Bit 0: Party line/MPI Bit 1: communication bus
kreis	1 byte	Maximum number of GD groups of the CPU
sk 1	1 byte	Maximum number of GD packets to be sent per GD circle of the CPU
sk 2	1 byte	Maximum number of GD packets to be sent for all GD circles of the CPU
ek 1	1 byte	Maximum number of GD packets to be received per GD circle of the CPU
ek 2	1 byte	Maximum number of GD packets for all GD circles of the CPU
len 1	1 byte	Maximum length of a GD packet
len 2	1 byte	Length of the GD packet header
len 3	1 byte	Length of the object description header
res	19 bytes	Reserved

## B.27 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0008

**Content** The partial list extract with SZL-ID W#16#0131 and the index W#16#0008 contains information about the time required for test and installation functions.

**Data Record** A data record of partial list extract SZL-ID W#16#0131 with index W#16#0008 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0008: Test and installation function time information
last 1	1 word	Basic overhead for status block*
last 2	1 word	Basic overhead for monitor variables*
last 3	1 word	Basic overhead for modify variables*
merker	1 word	Time for one variable address “memory bit”*
ea	1 word	Time for one variable address “input” or “output”
tz	1 word	Time for one variable address “timer” or “counter”
db	1 word	Time for one variable address “data block DB”
ld	1 word	Time for one variable address “ADB” or “local data”
reg	1 word	Time for one variable address “register”
ba_stali1	1 word	Basic time for a status list ID of group 1
ba_stali2	1 word	Basic time for a status list ID of group 2
ba_stali3	1 word	Basic time for a status list ID of group 3
akku	1 word	Accumulators added to basic time when ACCU 1, 2 addressed
adress	1 word	Address register added to basic time, when AR 1 or AR 2 addressed
dbreg	1 word	DB register added to basic time when DB register addressed
res	4 words	Reserved

\* Format: Bit 0 to bit 11 contains the time value (0 to 4K-1); bit 12 to 15 contains the time base; time base: 0 = 10<sup>-10</sup>sec, 1 = 10<sup>-9</sup>sec, A<sub>H</sub> = 10<sup>0</sup>sec, F = 10<sup>5</sup>sec

## B.28 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0009

### Content

The partial list extract with SZL-ID W#16#0131 and the index W#16#009 contains time-of-day capability parameters.

### Data Record

A data record of the partial list extract with SLZ-ID W#16#0131 and the index W#16#0009 has the following structure:

Name	Length	Meaning
index	1 word	W#16#0009: time of day
snyc_k	1 byte	Bits for time-of-day synchronization C bus Bit 0 : time-of-day synchronization neutral Bit 1: capable of being slave for time-of-day synchronization Bit 2: capable of being master for time-of-day synchronization
sync_mpi	1 byte	Bit pattern for time-of-day synchronization via MPI Bit assignment as for sync_k
sync_mfi	1 byte	Bit pattern for time-of-day synchronization via MFI, Bit assignment as for sync_k
res1	1 byte	Reserved
abw_puf	1 word	Clock deviation in ms/day when backed up
abw_5V	1 word	Clock deviation in ms/day in 5 V operation
anz_bsz	1 word	Number of run-time meters
res2	28 bytes	Reserved

## B.29 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0010

**Content** The partial list extract with SZL-ID W#16#0131 and index W#16#010 contains message parameters.

**Data Record** A data record of the partial list extract with SLZ-ID W#16#0131 and the index W#16010 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0010: message parameter
funk_1	1 byte	Bit pattern of the available S7 message functions Bit 0: Group status messages exist Bit 1: SCAN possible Bit 2: NOTIFY, ALARM, ALARM_8P, ALARM_8, (multicast) possible Bit 3: Sending archive data possible Bit 4-7: Reserve
funk_2	1 byte	Reserved
ber_meld_1	1 byte	Permitted address areas for messages (SCAN) Bit 0=1: PII Bit 1=1: PIQ Bit 2=1: M Bit 3=1: DB Bit 4-7: Reserved
ber_meld_2	1 byte	Reserved
ber_zus_1	1 byte	Permitted address areas additional values (SCAN) Bit 0=1: PII Bit 1=1: PIQ Bit 2=1: M Bit 3=1: DB Bit 4=1: T Bit 5=1: C Bit 6-7: Reserved
ber_zus_2	1 byte	Reserved
typ_zus_1	1 byte	Permitted data types for additional values (SCAN) Bit 0=1: Bit Bit 1=1: Byte Bit 2=1: Word Bit 3=1: DWord Bit 4=1: Timer Bit 5=1: Counter Bit 6=1: ARRAY OF CHAR[16] Bit 7 Reserve
typ_zus_2	1 byte	Reserved
maxanz_arch	1 word	Maximum number of archives for "Send Archive"
res	14 words	Reserved

## B.30 Data Record of the Partial List Extract with SZL-ID W#16#0131 Index W#16#0011

### Content

The partial list extract with the SZL-ID W#16#0131 and index W#16#0011 contains information about the SCAN functions.

### Data Record

A data record of the partial list extract with SLZ-ID W#16#0131 and the index W#16011 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0011: "SCAN"
funkt_0 to funkt_1	2 bytes	Available SCANS (bit = 1: available)
funkt_0	1 byte	Bit 0: SCAN1, bit 1: SCAN2, bit 2: SCAN3, bit 3: SCAN4, bit 4: SCAN5, bit 5 to bit 7: reserved
funkt_1	1 byte	Reserved
max_mel	1 word	Maximum number of configurable messages
max_mel_1	1 word	Maximum number of messages in SCAN1
zykl_1	1 word	Cycle time in SCAN1 in ms
max_zus_1	1 word	Maximum number of additional values per message in SCAN1
max_mel_2	1 word	Maximum number of messages in SCAN2
zykl_2	1 word	Cycle time in SCAN2 in ms
max_zus_2	1 word	Maximum number of additional values per message in SCAN2
max_mel_3	1 word	Maximum number of messages in SCAN3
zykl_3	1 word	Cycle time in SCAN3 in ms
max_zus_3	1 word	Maximum number of additional values per message in SCAN3
max_mel_4	1 word	Maximum number of messages in SCAN4
zykl_4	1 word	Cycle time in SCAN4 in ms
max_zus_4	1 word	Maximum number of additional values per message in SCAN4
max_mel_5	1 word	Maximum number of messages in SCAN5
zykl_5	1 word	Cycle time in SCAN5 in ms (0: acyclic SCAN)
max_zus_5	1 word	Maximum number of additional values per message in SCAN5
res	2 words	Reserved

## B.31 SZL-ID W#16#xy32 – Communication Status Data

**Purpose** If you read the partial list with SZL-ID W#16#xy32 you obtain the status data of module communication.

**Header** The header of partial list SZL-ID W#16#xy32 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0132 Status data for one communication section of the CPU You specify the communication section of the CPU with the INDEX parameter. W#16#0F32 Only partial list header information
INDEX	Only for SZL-ID W#16#0132: Communication section of the CPU W#16#0001 General data for communication W#16#0002 Test and installation status W#16#0003 Operator interface status W#16#0004 Object management system status W#16#0005 Diagnostics W#16#0006 Data exchange with CFBs W#16#0007 Global data W#16#0008 Time system W#16#0009 MPI status W#16#000A Communication bus status W#16#0010 S7-SCAN part 1 W#16#0011 S7-SCAN part 2
LENGTHDR	W#16#00028: one data record is 20 words long (40 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#??32 is always 20 words long. The data records have different contents. The content depends on the INDEX parameter, in other words, on the communication section of the CPU to which the data record belongs.



## B.32 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0001

### Content

The partial list extract with SZL-ID W#16#0132 and index W#16#0001 contains general communication status data.

### Data Record

A data record of partial list extract SZL-ID W#16#0132 with index W#16#0001 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0001: General status data for communication
res pg	1 word	Guaranteed number of PG connections
res os	1 word	Guaranteed number of OS connections
u pg	1 word	Current number of PG connections
u os	1 word	Current number of OS connections
proj	1 word	Current number of configured connections
auf	1 word	Current number of connections established by proj
free	1 word	Number of free connections
used	1 word	Number of free connections used
last	1 word	Maximum selected communication load of the CPU in %
res	10 words	Reserved

### **B.33 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0002**

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0002 contains information about the test and installation function status of the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0002 has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#0002: Test and installation status
anz	1 word	Number of initialized test and installation jobs
res	18 words	Reserved

### **B.34 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0003**

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0002 contains information about the test and installation function status of the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0003 has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#0002: Test and installation status
anz	1 word	Number of initialized test and installation jobs
res	18 words	Reserved

### B.35 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0004

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0004 contains information about the protection level of the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0004 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0004: Protection status data
key	1 word	Protection level for the key switch (possible values: 1,2 or 3)
param	1 word	Assigned protection level (possible values: 0, 1, 2 or 3; 0 means: no password assigned, assigned protection level is not valid)
real	1 word	Valid protection level of the CPU (possible values: 1, 2 or 3)
bart_sch	1 word	Position of the mode switch 0: undefined or cannot be ascertained 1: RUN 2: RUN_P 3: STOP 4: MRES
crst_wrst	1 word	Setting of the CRST/WRST switch 0: undefined, does not exist or cannot be be ascertained 1: CRST 2: WRST
res	14 words	Reserved

### B.36 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0005

#### Content

The partial list extract with SZL-ID W#16#0132 and index W#16#0005 contains information about the status of the diagnostics on the module.

#### Data Record

A data record of partial list extract SZL-ID W#16#0132 with index W#16#0005 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0005: Diagnostics
erw	1 word	Extended functions 0: no 1: yes
send	1 word	Automatic sending 0: no 1: yes
moeg	1 word	Sending user-defined diagnostic messages currently possible 0: no 1: yes
ltmerz	1 word	Generation of status message active 0: no 1: yes
res	15 words	Reserved

### B.37 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0006

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0006 contains status data about data exchange with communication SFBs for configured connections.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0006 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0006: Data exchange with communication SFBs for configured connections
used	8 bytes	Blocks used, structure see Section B.25
anz_schnell	1 byte	Reserved
anz_inst	1 word	Number of loaded SFB instances
anz_multicast	1 word	Number of blocks used for multicast
res	25 bytes	Reserved

### **B.38 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0007**

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0007 contains information about the GD status (global data).

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0007 has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#0007: GD status data
anz	1 word	Number of the existing GD packets
list	1 word	List of used GD circuits
res	14 words	Reserved

### B.39 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0008

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0008 contains information about the status of the time system on the module.

**Data Record** A data record of partial list extract SZL-ID W#16#01032 with index W#16#0008 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0008: Time system status
zykl	1 word	Cycle time of the synchronization frames
korrr	1 word	Correction factor for the time
clock 0	1 word	Run-time meter 0: time in hours
clock 1	1 word	Run-time meter 1: time in hours
clock 2	1 word	Run-time meter 2: time in hours
clock 3	1 word	Run-time meter 3: time in hours
clock 4	1 word	Run-time meter 4: time in hours
clock 5	1 word	Run-time meter 5: time in hours
clock 6	1 word	Run-time meter 6: time in hours
clock 7	1 word	Run-time meter 7: time in hours
time	4 words	Current date and time (format: DATE_AND_TIME)
bszl_0 to bszl_1	1 word	Run-time meter active (bit =1: run-time meter active)
bszl_0	1 byte	Bit x: run-time meter x, $0 \leq x \leq 7$
bszl_1	1 byte	Reserved
bszü_0 to bszü_1	1 word	Run-time meter overflow (bit = 1: overflow)
bszü_0	1 byte	Bit x: run-time meter x, $0 \leq x \leq 7$
bszü_1	1 byte	Reserved
res	3 words	Reserved



## **B.40 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0009**

**Content**                      The partial list extract with SZL-ID W#16#0132 and index W#16#0009 contains information about the MPI.

**Data Record**                A data record of partial list extract SZL-ID W#16#0132 with index W#16#0009 has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#0009: MPI status
bps	2 words	Data rate used (hexadecimal coded)
res	17 words	Reserved

## **B.41 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#000A**

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#000A contains information about the communication bus used by the module.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#000A has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#000A: Index for communication bus
mpi_Bps	2 words	Data rate used (hexadecimal coded)
res	17 words	Reserved

## B.42 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0010

### Content

The partial list extract with SZL-ID W#16#0132 and index W#16#0010 contains information about the first four SCAN cycles. If any of the SDBs configured for one of these cycles do not exist, the current number of configured messages for this SCAN cycle is zero and the time stamp is irrelevant.

### Data Record

A data record of partial list extract SZL-ID W#16#0132 with index W#16#0010 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0010: SCAN part 1
anz_scan1	1 word	Number of configured SCAN1 messages
zeit_1	3 words	Time stamp of the SDBs for SCAN1
anz_scan2	1 word	Number of configured SCAN2 messages
zeit_2	3 words	Time stamp of the SDBs for SCAN2
anz_scan3	1 word	Number of configured SCAN3 messages
zeit_3	3 words	Time stamp of the SDBs for SCAN3
anz_scan4	1 word	Number of configured SCAN4 messages
zeit_4	3 words	Time stamp of the SDBs for SCAN4
res	3 words	Reserved

### **B.43 Data Record of the Partial List Extract with SZL-ID W#16#0132 Index W#16#0011**

**Content** The partial list extract with SZL-ID W#16#0132 and index W#16#0011 contains information about the fifth SCAN cycle.

**Data Record** A data record of partial list extract SZL-ID W#16#0132 with index W#16#0011 has the following structure:

<b>Name</b>	<b>Length</b>	<b>Meaning</b>
Index	1 word	W#16#0011: SCAN part 2
anz_scan	1 word	Number of configured SCAN5 messages
zeit	3 words	Time stamp of the SDBs for SCAN5
res	15 words	Reserved

## B.44 SZL-ID W#16#xy33 – Stations for S7 Messages and Diagnostic Events

**Purpose** If you read the partial list SZL-ID W#16#xy33, you obtain information about the devices logged on with the module.

**Header** The header of partial list SZL-ID W#16#xy33 is structured as follows:

	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0033: All stations logged on for messages and diagnostic events W#16#0F33: Only partial list header information
INDEX	Always W#16#0000
LENGTHDR	W#16#000A: one data record is 5 words long (10 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xy33 has the following structure:

Name	Length	Meaning
name	4 words	User name of the logged-on station
code_0 to code_1	2 bytes	Function ID (bit = 0: no logon for this message class, bit = 1: logon for this message class exists)
code_0	1 byte	Bit = 0: operating system messages (VMD status) bit 1: system diagnostic events, bit 2: user-defined diagnostic events, bit 3: group status messages, bit 4: block-related messages (NOTIFY, ALARM, ALARM_8P, ALARM_8), bit 5: symbol-related messages (SCAN), bit 6 to 7: 0
code_1	1 byte	Bit = 0 to bit 2: 0 bit 3: send archive data (bit = 0: no logon, bit = 1: logon exists), bit 4: block-related messages (ALARM_SQ, ALARM_S), bit 5 to bit 7: 0

## B.45 SZL-ID W#16#xy81 – Local Data of the OBs

### Purpose

If you read the partial list SZL-ID W#16#xy81, you obtain startup information about the OBs of the module currently being executed or still to be executed. The startup information is located in the first 20 bytes of the local data of an OB.

### Header

The header of partial list SZL-ID W#16#xy81 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0081 Startup information of all OBs W#16#0181 Startup information of all synchronous error OBs W#16#0281 Startup information of all synchronous error OBs of one priority class W#16#0381 Startup information of all OBs of one priority class W#16#0581 Startup information of all synchronous error OBs before processing W#16#0681 Startup information of all synchronous error OBs of a priority class before processing W#16#0781 Startup information of all OBs of one priority class before processing W#16#0881 Startup information of all OBs before processing W#16#0981 Startup information of all synchronous error OBs being processed W#16#0A81 Startup information of all synchronous error OBs of a priority class being processed W#16#0B81 Startup information of all OBs of one priority class being processed W#16#0C81 Startup information of all OBs being processed W#16#0F81 Only partial list header
INDEX	Priority class
LENGTHDR	W#16#0014: one data record is 10 words long (20 bytes)
N_DR	Number of data records

### Data Record

A data record of partial list extract SZL-ID W#16#0081 is always 10 words (20 bytes) long. The content of the local data is described in Chapter 1.

## B.46 SZL-ID W#16#xy82 – Startup Events

**Purpose** If you read the partial list SZL-ID W#16#xy82, you obtain a short version of the startup information about the OBs of the module currently being executed or still to be executed.

**Header** The header of partial list SZL-ID W#16#xy82 is structured as follows:

Content	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0082 All startup events W#16#0182 Startup events of all current synchronous error OBs W#16#0282 Startup events of all current synchronous error OBs of a priority class W#16#0382 Startup events of all OBs of a priority class W#16#0582 Startup events of all synchronous error OBs before processing W#16#0682 Startup events of all synchronous error OBs of a priority class before processing W#16#0782 Startup events of all OBs of a priority class before processing W#16#0882 Startup events of all OBs before processing W#16#0982 Startup events of all synchronous error OBs being processed W#16#0A82 Startup events of all synchronous error-OBs of a priority class being processed W#16#0B82 Startup events of all OBs of a priority class being processed W#16#0C82 Startup events of all OBs being processed W#16#0f82 Only partial list nkopfinfo
INDEX	Number of the priority class
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list extract SZL-ID W#16#xy82 has the following structure:

Name	Length	Meaning
ereig	1 word	Event ID
ae	1 byte	Priority class
ob	1 byte	OB number

## B.47 SZL-ID W#16#xy91 – Module Status Information

**Purpose** If you read the partial list SZL-ID W#16#xy91, you obtain the status information of modules.

**Header** The header of partial list SZL-ID W#16#xy91 is structured as follows:

Content	Meaning
SZL-ID	<p>The SZL-ID of the partial list extract</p> <p>W#16#0091 Module status information of all plugged in modules and submodules</p> <p>W#16#0191 Status information of all modules/racks with wrong type identifier</p> <p>W#16#0291 Status information of all faulty modules</p> <p>W#16#0391 Status information of all modules that are not available</p> <p>W#16#0591 Status information of all submodules of the host module</p> <p>W#16#0991 Module status information of a DP master system</p> <p>W#16#0A91 Module status information of all DP master systems</p> <p>W#16#0C91 Status information of a module in the central rack or connected to an integrated DP communications processor via the logical base address</p> <p>W#16#4C91 Status information of a module connected to an external DP communications processor via the logical base address</p> <p>W#16#0D91 Module status information of all modules in the specified rack/in the specified station (DP)</p> <p>W#16#0E91 Module status information of all configured modules</p> <p>W#16#0F91 Only partial list header information</p>
INDEX	<ul style="list-style-type: none"> <li>For the partial list extract with SZL-ID W#16#0C91: <ul style="list-style-type: none"> <li>S7-400: bit 0 to 14: logical base address of the module bit 15: 0 = input, 1 = output</li> <li>S7-300: module start address</li> </ul> </li> <li>For the partial list extract with SZL-ID W#16#4C91 (only S7-400): <ul style="list-style-type: none"> <li>Bits 0 to 14 : logical base address of the module</li> <li>Bit 15 : 0 = input, 1 = output</li> </ul> </li> <li>For the partial list extract with SZL-IDs W#16#0091, W#16#0191, W#16#0291, W#16#0391, W#16#0491, W#16#0591, W#16#0A91, W#16#0E91, W#16#0F91: INDEX is irrelevant, all modules (in the rack and in the distributed I/Os)</li> <li>For the partial list extract with SZL-IDs W#16#0991 and W#16#0D91: <ul style="list-style-type: none"> <li>W#16#00xx all modules and submodules of a rack (xx contains the number of the rack)</li> <li>W#16#xx00 all modules of a DP master system (xx contains the DP master system ID)</li> <li>W#16#xxyy all modules of a DP station (xx contains the master system ID, yy station number)</li> </ul> </li> </ul>



Content	Meaning
LENGTHDR	W#16#0010: one data record is 8 words long (16 bytes)
N_DR	Number of data records

**Data Record**

A data record of partial list ID W#16#xy91 has the following structure:

Name	Length	Meaning
adr1	1 word	Number of the rack (DP master system ID and station number with DP) (geographical address)
adr2	1 word	Slot and submodule slot
logadr	1 word	First assigned logical I/O address (base address)
solltyp	1 word	Expected module type
isttyp	1 word	Actual module type
alarm	1 word	Reserved
eastat	1 word	I/O status Bit 0 = 1: module fault (detected by diagnostic interrupt) Bit 1 = 1: module exists Bit 2 = 1: module does not exist (detected as access error) Bit 5 = 1: module can be host module for submodule Bit 6 = 1: reserved for S7-400 Bit 7 = 1: module on local bus segment Bit 8 to bit 15: data ID for logical address (input: B#16#B4, output: B#16#B5)
ber_bgr	1 word	Area ID/module width Bit 0 to bit 2 : module width Bit 4 to bit 6 : area ID 0 = S7-400 1 = S7-300 2 = ET area 3 = P area 4 = Q area 5 = IM3 area 6 = IM4 area

**adr1**

The parameter adr1 contains the following:

- when installed centrally, the rack number.

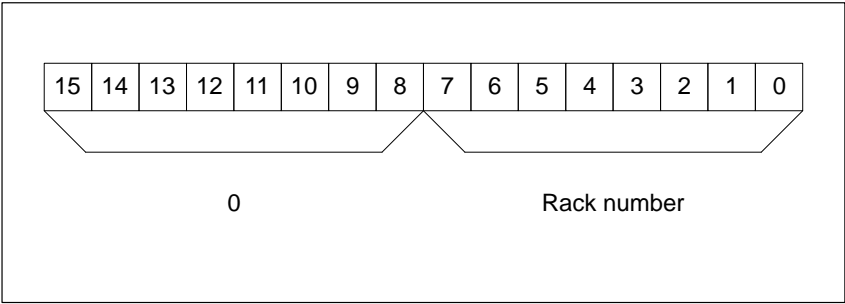


Figure B-2 Bits of the Parameter adr1 when Installed Centrally

- with a distributed configuration
  - the DP master system ID
  - the station number.

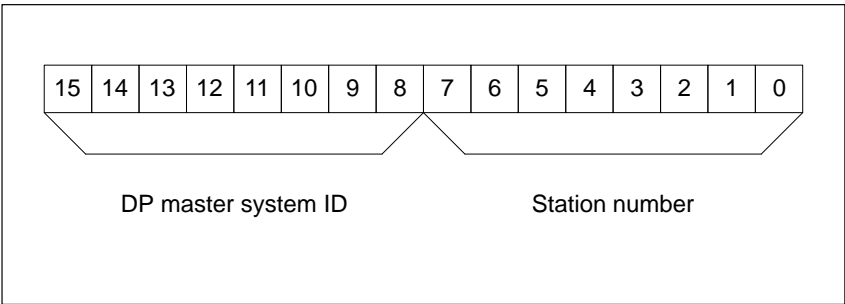


Figure B-3 Bits of the Parameter adr1 in a Distributed Configuration

**adr2**

The parameter adr2 contains the slot and submodule slot.

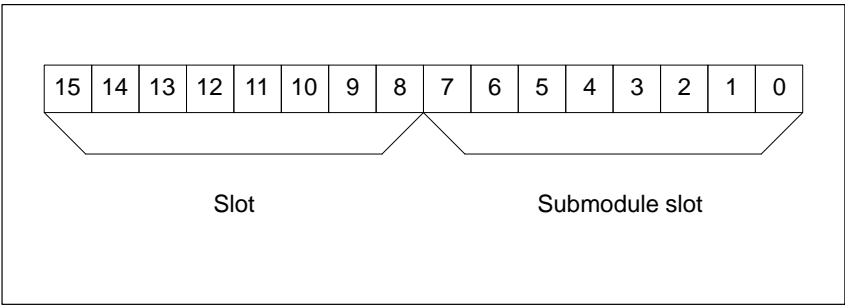


Figure B-4 Bits of the Parameter adr2

## B.48 SZL-ID W#16#xy92 – Rack / Station Status Information

**Purpose** If you read the partial list SZL-ID W#16#xy92, you obtain information about the expected and the current hardware configuration of centrally installed racks and stations of a DP master system.

**Header** The header of partial list SZL-ID W#16#xy92 is structured as follows:

Content	Meaning
SZL-ID	<p>The SZL-ID of the partial list extract:</p> <p>W#16#0092: expected status of the central racks/stations of a DP master system connected via an integrated DP interface</p> <p>W#16#4092: expected status of the stations of a DP master system connected via an external DP interface</p> <p>W#16#0292: actual status of the central racks/stations of a DP master system connected via an integrated DP interface</p> <p>W#16#4292: actual status of the stations of a DP master system connected via an external DP interface</p> <p>W#16#0392: state of the battery backup of the racks in a central configuration</p> <p>W#16#0492: state of the total backup of the racks in a central configuration</p> <p>W#16#0592: state of the 24 V power supply of the modules in a central configuration</p> <p>W#16#0692: OK state of the expansion racks in the central configuration / of the stations of a DP master system connected via an integrated DP interface</p> <p>W#16#4692: OK state of the stations of a DP master system connected via an external DP interface</p> <p>W#16#0F92: only partial list header information of the “0x92” list</p> <p>W#16#4F92: only partial list header information of the “4x92” list</p>
INDEX	0/ DP master system ID
LENGTHDR	W#16#0010: one data record is 8 words long (16 bytes)
N_DR	Number of data records

**Data Record**

A data record of the partial list with the ID W#16#xy92 has the following structure:

Content	Length	Meaning
status_0 to status_15	16 bytes	<p>Rack status/ station status or backup status.</p> <p>W#16#0092: Bit=0: rack/station not configured Bit=1: rack/station configured</p> <p>W#16#4092 Bit=0: station not configured Bit=1: station configured</p> <p>W#16#0292: Bit=0: rack/station failure or not configured Bit=1: rack/station exists and has not failed</p> <p>W#16#4292: Bit=0: station failure or not configured Bit=1: station exists and has not failed</p> <p>W#16#0392: Bit=0: no battery failed in rack/station Bit=1: at least one battery must be replaced in rack/station</p> <p>W#16#0492: Bit=0: backup voltage exists Bit=1: no backup voltage</p> <p>W#16#0592: Bit=0: 24V power supply exists Bit=1: no 24V power supply</p> <p>W#16#0692: Bit=0: all modules of the expansion rack/of a station exist, are available and no problems Bit=1: at least 1 module of the expansion rack/of a station is not OK</p> <p>W#16#4692: Bit=0: all modules of a station exist are available and no problems Bit=1: at least 1 module of a station is not ok</p>
status_0	1 byte	<p>Bit 0: Central rack (INDEX = 0) or station 1 (INDEX &lt; &gt; 0)</p> <p>Bit 1: 1. Expansion rack or station 2 : : : Bit 7: 7. Expansion rack or station 8</p>
status_1	1 byte	<p>Bit 0: 8. Expansion rack or station 9 : : : Bit 7: 15. Expansion rack or station 16</p>
status_2	1 byte	<p>Bit 0: 16. Expansion rack or station 17 : : : Bit 5: 21. Expansion rack or station 22 Bit 6: 0 or station 23 Bit 7: 0 or station 24</p>

Content	Length	Meaning
status_3	1 byte	Bit 0: 0 or station 25 : : Bit 5: 0 or station 30 Bit 6: Expansion rack (SIMATIC S5 area) or station 31 Bit 7: 0 or station 32
status_4	1 byte	Bit 0: 0 or station 33 : : Bit 7: 0 or station 40
:		
:		
status_15	1 byte	Bit 0: 0 or station 121 : : Bit 7: 0 or station 128

## B.49 SZL-ID W#16#xyA0 – Diagnostic Buffer

**Purpose** If you read the partial list SZL-ID W#16#xyA0, you obtain the entries in the diagnostic buffer of the module.

**Header** The header of partial list SZL-ID W#16#xyA0 is structured as follows:

Content	Meaning
SZL-ID	<p>The SZL-ID of the partial list extract:</p> <p>W#16#00A0: All entries possible in the current mode</p> <p>W#16#01A0: The most recent entries; you specify the number of most recent entries with the INDEX parameter.</p> <p>W#16#04A0: Start information of all standard OBs</p> <p>W#16#05A0: All entries from communications units</p> <p>W#16#06A0: All entries of the object management system</p> <p>W#16#07A0: All entries of the test and installation function</p> <p>W#16#08A0: All entries due to operating statuses</p> <p>W#16#09A0: All entries caused by asynchronous errors</p> <p>W#16#0AA0: All entries caused by synchronous errors</p> <p>W#16#0BA0: All entries caused by STOP, abort, mode transition</p> <p>W#16#0CA0: All entries caused by fault-tolerant/fail-safe events</p> <p>W#16#0DA0: All diagnostic entries</p> <p>W#16#0EA0: All user entries</p> <p>W#16#0FA0: Only partial list header information</p>
INDEX	<p>Only for SZL-ID W#16#01A0:</p> <p>Number of most recent entries</p>
LENGTHDR	W#16#0014: one data record is 10 words long (20 bytes)
N_DR	Number of data records

**Data Record** A data record of partial list SZL-ID W#16#xyA0 has the following structure:

Name	Length	Meaning
ID	1 word	Event ID
info	5 words	Information about the event and its consequences
time	4 words	Time stamp of the event

**Diagnostic Buffer** You obtain more information about the events in the diagnostic buffer using STEP 7.

## B.50 SZL-ID W#16#00B1 – Module Diagnostic Information

### Purpose

If you read the partial list SZL-ID W#16#00B1, you obtain the first 4 diagnostic bytes of a module with diagnostic capability.

### Header

The header of partial list SZL-ID W#16#00B1 is structured as follows:

	Meaning
SZL-ID	W#16#00B1
INDEX	Bit 0 to bit 14: logical base address Bit 15: 0 = input, 1 = output
LENGTHDR	W#16#0004: one data record is 2 words long (4 bytes)
N_DR	1

### Data Record

A data record of partial list SZL-ID W#16#00B1 has the following structure:

Name	Length	Meaning
byte1	1 byte	Bit 0: Module fault/OK (group fault ID) Bit 1: Internal fault Bit 2: External fault Bit 3: Channel error exists Bit 4: No external auxiliary voltage Bit 5: No front connector Bit 6: Module not assigned parameters Bit 7: Wrong parameters on module
byte2	1 byte	Bit 0 to bit 3: Module class (CPU, FM, CP, IM, SM, ...) Bit 4: Channel information exists Bit 5: User information exists Bit 6: Diagnostic interrupt from substitute Bit 7: Reserve (initialized with 0)
byte3	1 byte	Bit 0: User module incorrect/does not exist Bit 1: Communication fault Bit 2: Mode RUN/STOP (0 = RUN, 1 = STOP) Bit 3: Watchdog responded Bit 4: Internal module power supply failed Bit 5: Battery exhausted (BFS) Bit 6: Entire buffer failed Bit 7: Reserve (initialized with 0)
byte4	1 byte	Bit 0: Expansion rack failure (detected by IM) Bit 1: Processor failure Bit 2: Eprom error Bit 3: RAM error Bit 4: ADC/DAC error Bit 5: Fuse blown Bit 6: Hardware error lost Bit 7: Reserve (initialized with 0)

## B.51 SZL-ID W#16#00B2 – Diagnostic Data Record 1 with Geographical Address

### Purpose

If you read the partial list with SZL-ID W#16#00B2, you obtain diagnostic data record 1 of a module in a central rack (not for DP or submodules). You specify the number using the rack and slot number.

### Header

The header of partial list SZL-ID W#16#00B2 is structured as follows:

Content	Meaning
SZL-ID	W#16#00B2
INDEX	W#16#xxyy: xx contains the number of the rack yy contains the slot number
LENGTHDR	The length of the data record depends on the module.
N_DR	1

### Data Record

The size of a data record of partial list SZL-ID W#16#00B2 and its contents depend on the particular module. For further information refer to **/70/**, **/101/** and to the manual describing the module concerned.



## B.52 SZL-ID W#16#00B3 – Module Diagnostic Data with Logical Base Address

**Purpose** If you read the partial list SZL-ID W#16#00B3, you obtain all the diagnostic data of a module. You can also obtain this information for DP and submodules. You select the module using its logical base address.

**Header** The header of partial list SZL-ID W#16#00B3 is structured as follows:

Content	Meaning
SZL-ID	W#16#00B3
INDEX	Bit 0 to bit 14: logical base address Bit 15: 0 = input, 1 = output
LENGTHDR	The length of the data record depends on the module.
N_DR	1

**Data Record** The size of a data record of partial list SZL-ID W#16#00B3 and its content depend on the particular module. For further information refer to /70/, /101/ and to the manual describing the module concerned.

### B.53 SZL-ID W#16#00B4 – Diagnostic Data of a DP Slave

#### Purpose

If you read the partial list SZL-ID W#16#00B4, you obtain the diagnostic data of a DP slave. This diagnostic data is structured in compliance with EN50 170 Volume 2, PROFIBUS. You select the module using the diagnostic address you configured.

#### Header

The header of partial list SZL-ID W#16#00B4 is structured as follows:

Content	Meaning
SZL-ID	W#16#00B4
INDEX	Configured diagnostic address of the DP slave
LENGTHDR	Length of a data record. The maximum length is 240 bytes. For standard slaves which have a diagnostic data length of more than 240 bytes up to a maximum of 244 bytes, the first 240 bytes are read and the overflow bit is set in the data.
N_DR	1

#### Data Record

A data record of partial list SZL-ID W#16#00B4 has the following structure:

Name	Length	Meaning
status1	1 byte	Station status1
status2	1 byte	Station status2
status3	1 byte	Station status3
stat_nr	1 byte	Master station number
ken_hi	1 byte	Vendor ID (high byte)
ken_lo	1 byte	Vendor ID (low byte)
....	....	Further diagnostic data specific to the particular slave

# Events

# C

## Chapter Overview

Section	Description	Page
C.1	Events and Event ID	C-2
C.2	Event Class 1 – Standard OB Events	C-3
C.3	Event Class 2 – Synchronous Errors	C-5
C.4	Event Class 3 – Asynchronous Errors	C-6
C.5	Event Class 4 – Stop and Abort Events	C-8
C.6	Event Class 5 – Mode Run-time Events	C-11
C.7	Event Class 6 – Communication Events	C-13
C.8	Event Class 8 – Diagnostic Events for Modules	C-15
C.9	Event Class 9 – Standard User Events	C-17
C.10	Event Classes A and B – Free User Events	C-19
C.11	Reserved Event Classes	C-20

## C.1 Events and Event ID

### Event

All events are numbered within the SIMATIC S7 programmable logic controller. This allows you to relate a message text to an event.

### Event ID

An event ID is assigned to every event. The event ID is structured as follows:

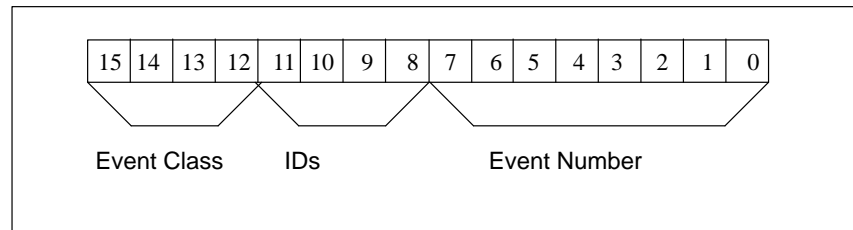


Figure C-1 Structure of the Event ID

### Event Class

The event classes are as follows:

Table C-1 Event Classes

Number	Event Class
1	Standard OB events
2	Synchronous errors
3	Asynchronous errors
4	Mode transitions
5	Run-time events
6	Communication events
7	Events for fail-safe and fault-tolerant systems
8	Standardized diagnostic data on modules
9	User-defined events
A, B	Freely definable events
C to F	Reserved

### Identifier

The identifier is used to distinguish the type of events. Events can be caused by the following:

- external errors and
- internal errors

## C.2 Event Class 1 – Standard OB Events

Event ID	Event
W#16#1101	OB 1 was started, initiated after complete restart
W#16#1102	OB 1 was started, initiated after restart
W#16#1103	OB 1 was started after end of OB1 (free cycle)
W#16#1111	Time-of-day interrupt comparator 1
W#16#1112	Time-of-day interrupt comparator 2
W#16#1113	Time-of-day interrupt comparator 3
W#16#1114	Time-of-day interrupt comparator 4
W#16#1115	Time-of-day interrupt comparator 5
W#16#1116	Time-of-day interrupt comparator 6
W#16#1117	Time-of-day interrupt comparator 7
W#16#1118	Time-of-day interrupt comparator 8
W#16#1121	Time-delay interrupt counter 1
W#16#1122	Time-delay interrupt counter 2
W#16#1123	Time-delay interrupt counter 3
W#16#1124	Time-delay interrupt counter 4
W#16#1131	Cyclic interrupt clock pulse generator 1
W#16#1132	Cyclic interrupt clock pulse generator 2
W#16#1133	Cyclic interrupt clock pulse generator 3
W#16#1134	Cyclic interrupt clock pulse generator 4
W#16#1135	Cyclic interrupt clock pulse generator 5
W#16#1136	Cyclic interrupt clock pulse generator 6
W#16#1137	Cyclic interrupt clock pulse generator 7
W#16#1138	Cyclic interrupt clock pulse generator 8
W#16#1139	Cyclic interrupt clock pulse generator 9
W#16#1141	Hardware interrupt interrupt line 1
W#16#1142	Hardware interrupt interrupt line 2
W#16#1143	Hardware interrupt interrupt line 3
W#16#1144	Hardware interrupt interrupt line 4
W#16#1161	Self-triggered multicomputing interrupt
W#16#1162	Externally triggered multicomputing interrupt
W#16#1381	Request for manual complete restart
W#16#1382	Request for automatic complete restart
W#16#1383	Request for manual restart
W#16#1384	Request for automatic restart

Event ID	Event
W#16#1191	OB 90 start event initiated by complete restart
W#16#1192	OB 90 start event initiated by deleting a block
W#16#1193	OB 90 start event initiated by loading OB 90 in the CPU RUN mode
W#16#1195	OB 90 start event (completion of background cycle)

### C.3 Event Class 2 – Synchronous Errors

Table C-2 Events of Event Class 2 – Synchronous Errors

Event ID	Event	OB
W#16#2521	BCD conversion error	OB 121
W#16#2522	Area length error when reading	
W#16#2523	Area length error when writing	
W#16#2524	Area error when reading	
W#16#2525	Area error when writing	
W#16#2526	Timer number error	
W#16#2527	Counter number error	
W#16#2528	Alignment error when reading	
W#16#2529	Alignment error when writing	
W#16#2530	Write error when accessing the DB	
W#16#2531	Write error when accessing the DI	
W#16#2532	Block number error when opening a DB	
W#16#2533	Block number error when opening a DI	
W#16#2534	Block number error when calling an FC	
W#16#2535	Block number error when calling an FB	
W#16#253A	DB not loaded	
W#16#253C	FC not loaded	
W#16#253D	SFC not loaded	
W#16#253E	FB not loaded	
W#16#253F	SFB not loaded	
W#16#2942	I/O access error, reading	OB122
W#16#2943	I/O access error, writing	
W#16#2944	I/O access error in the nth read access ( $n > 1$ )	
W#16#2945	I/O access error in the nth write access ( $n > 1$ )	

## C.4 Event Class 3 – Asynchronous Errors

Event ID	Event	OB
W#16#3501	Watchdog monitoring (cycle time exceeded)	OB80
W#16#3502	User interface (OB or FRB) request error	
W#16#3505	Time-of-day interrupt(s) skipped due to new clock setting	
W#16#3506	Time-of-day interrupt(s) skipped when changing to RUN after HOLD	
W#16#3507	Multiple OB request errors caused start information buffer overflow	
W#16#3921/3821	BATTF: at least one backup battery of the central rack exhausted/ problem eliminated	OB81
W#16#3922/3822	BAF: no backup voltage on central rack/ problem eliminated	
W#16#3923/3823	24 volt supply failure on central rack / problem eliminated	
W#16#3931/3831	BATTF: at least one backup battery on at least one expansion rack exhausted/ problem eliminated	
W#16#3932/3832	BAF: no backup voltage on at least one of the expansion racks/ problem eliminated	
W#16#3933/3833	24 volt supply failure on at least one expansion rack/ problem eliminated	OB82
W#16#3942	Module fault present	
W#16#3842	Module OK	OB83
W#16#3861	Module inserted, module type OK	
W#16#3961	Module removed, cannot be addressed	
W#16#3863	Module plugged in, but wrong module type	
W#16#3864	Module plugged in, but causing problem (type ID unreadable)	
W#16#3865	Module plugged in, but error in module parameter assignment	OB84
W#16#3981	Interface error entering state	
W#16#3881	Interface error leaving state	OB85
W#16#35A1	User interface (OB or FRB) not found	
W#16#35A2	OB not loaded (started by SFC or operating system due to configuration)	
W#16#35A3	Error when operating system accesses a block	
W#16#39B1	I/O access error when updating the process image input table	
W#16#39B2	I/O access error when transferring the process image to the output modules	



Event ID	Event	OB
W#16#38C1	Expansion rack failure (1 to 21), leaving state	OB86
W#16#39C1	Expansion rack failure (1 to 21), entering state	
W#16#38C2	Expansion rack operational again but mismatch between setpoint and actual configuration	
W#16#39C3	Distributed I/Os: master system failure entering state	
W#16#39C4	Distributed I/Os: station failure, entering state	
W#16#38C4	Distributed I/Os: station failure, leaving state	
W#16#39C5	Distributed I/Os: station fault, entering state	
W#16#38C5	Distributed I/Os: station fault, leaving state	
W#16#38C6	Expansion rack operational again, but error(s) in module parameter assignment	
W#16#38C7	Distributed I/Os: station operational again, but error(s) in module parameter assignment	
W#16#35D2	Diagnostic entries cannot be sent at present	OB87
W#16#35D3	Synchronization frames cannot be sent	
W#16#35D4	Illegal time jump resulting from synchronization	
W#16#35D5	Error adopting the synchronization time	
W#16#35E1	Incorrect frame ID in GD	
W#16#35E2	GD packet status cannot be entered in DB	
W#16#35E3	Frame length error in GD	
W#16#35E4	Illegal GD packet number received	
W#16#35E5	Error accessing DB in CFB	
W#16#35E6	GD total status cannot be entered in DB	

## C.5 Event Class 4 – Stop and Abort Events

Event ID	Event
W#16#4300	Backed-up power on
W#16#4301	Mode transition from STOP to STARTUP
W#16#4302	Mode transition from STARTUP to RUN
W#16#4303	STOP caused by stop switch being activated
W#16#4304	STOP caused by PG STOP operation or by SFB20 “STOP”
W#16#4305	HOLD: breakpoint reached
W#16#4306	HOLD: breakpoint exited
W#16#4307	Memory reset started by PG operation
W#16#4308	Memory reset started by switch setting
W#16#4309	Memory reset started automatically (power on not backed up)
W#16#430A	HOLD exited, transition to STOP
W#16#410B	Transition from STOP (Init) to STOP (internal)
W#16#410C	Transition from STOP (internal) to STOP (Init)
W#16#430D	STOP caused by other CPU in multicomputing
W#16#430E	Memory reset executed
W#16#4520	DEFECT: STOP not possible
W#16#4521	DEFECT: failure of MC5-ASIC
W#16#4522	DEFECT: failure of clock chip
W#16#4523	DEFECT: failure of clock pulse generator
W#16#4524	DEFECT: failure of timer update function
W#16#4525	DEFECT: failure of multicomputing synchronization
W#16#4926	DEFECT: failure of the watchdog for I/O access
W#16#4527	DEFECT: failure of I/O access monitoring
W#16#4528	DEFECT: failure of scan time monitoring
W#16#4530	DEFECT: memory test error in internal memory
W#16#4931	DEFECT: memory test error in memory submodule
W#16#4532	DEFECT: failure of core resources
W#16#4933	DEFECT: checksum error
W#16#4934	DEFECT: memory not available
W#16#4935	DEFECT: cancelled by watchdog/processor exceptions
W#16#4536	DEFECT: switch defective
W#16#4541	STOP caused by priority class system
W#16#4542	STOP caused by object management system
W#16#4543	STOP caused by test functions

Event ID	Event
W#16#4544	STOP caused by diagnostic system
W#16#4545	STOP caused by communication system
W#16#4546	STOP caused by CPU memory management
W#16#4547	STOP caused by process image management
W#16#4548	STOP caused by I/O management
W#16#4949	STOP caused by continuous hardware interrupt
W#16#454A	STOP caused by configuration, deselected OB was loaded during complete restart
W#16#494D	STOP caused by I/O error
W#16#494E	STOP caused by power failure
W#16#494F	STOP caused by configuration error
W#16#4550	DEFECT: internal system error
W#16#4555	No restart possible, monitoring time elapsed
W#16#4556	STOP: memory reset request from communication system
W#16#4357	Module watchdog started
W#16#4358	All modules are ready for operation
W#16#4959	STOP: one or more modules not ready for operation
W#16#4562	STOP caused by programming error (OB not loaded or not possible, or no FRB)
W#16#4563	STOP caused by I/O access (OB not loaded or not possible, or no FRB)
W#16#4568	STOP caused by time error (OB not loaded or not possible, or no FRB)
W#16#456A	STOP caused by diagnostic interrupt (OB not loaded or not possible, or no FRB)
W#16#456B	STOP caused by removing/inserting module (OB not loaded or not possible, or no FRB)
W#16#456C	STOP caused by CPU hardware error (OB not loaded or not possible, or no FRB)
W#16#456D	STOP caused by program sequence error (OB not loaded or not possible, or no FRB)
W#16#456E	STOP caused by communication error (OB not loaded or not possible, or no FRB)
W#16#456F	STOP caused by rack failure OB (OB not loaded or not possible, or no FRB)
W#16#4571	STOP caused by nesting stack error
W#16#4572	STOP caused by master control relay stack error
W#16#4573	STOP caused by exceeding the nesting depth for synchronous errors
W#16#4574	STOP caused by to exceeding interrupt stack nesting depth in the priority class stack

Event ID	Event
W#16#4575	STOP caused by exceeding block stack nesting depth in the priority class stack
W#16#4576	STOP caused by error when allocating the local data
W#16#4578	STOP caused by unknown opcode
W#16#457A	STOP caused by code length error
W#16#457B	STOP caused by DB not being loaded on on-board I/Os
W#16#457F	STOP caused by STOP command
W#16#4580	STOP: back-up buffer contents inconsistent (no transition to RUN)
W#16#4590	STOP caused by overloading the internal functions
W#16#49A0	STOP caused by parameter assignment error: startup disabled (reload parameter assignment information)
W#16#49A1	STOP caused by parameter assignment error: memory reset request
W#16#49A2	STOP caused by error in parameter modification: startup disabled
W#16#49A3	STOP caused by error in parameter modification: memory reset request
W#16#49A4	STOP: inconsistency in configuration data
W#16#49A5	STOP: distributed I/Os: inconsistency in the loaded configuration information
W#16#49A6	STOP: distributed I/Os: invalid configuration information
W#16#49A7	STOP: distributed I/Os: no configuration information
W#16#49A8	STOP: error indicated by the interface module for the distributed I/Os
W#16#43B0	Firmware update was successful
W#16#49B1	Firmware update data incorrect
W#16#49B2	Firmware update: hardware version does not match firmware
W#16#49B3	Firmware update: module type does not match firmware

## C.6 Event Class 5 – Mode Run-Time Events

Event ID	Event
W#16#5101	Initialization of communication (evaluation of the module card OK)
W#16#5102	Start initialization without memory reset (battery backed-up POWER UP)
W#16#5105	Initialization: CPU hardware tests
W#16#5107	Data consistency check
W#16#5109	Checksum
W#16#510B	Start of initialization after memory reset
W#16#510C	Start of stop loop (CPU capable of communication)
W#16#530D	New startup information in the STOP mode
W#16#510E/500E	Wait point 1 "exit STOP" reached/passed
W#16#5310	IM wait time expired
W#16#5111	Startup although ready message not received from modules
W#16#5122	Start of complete restart system activity
W#16#5123	End of communication bus parameter assignment
W#16#5124	Start of restart system activity
W#16#5132/5032	Wait point "begin start-up OB processing" reached/passed
W#16#5134	Return to STARTUP mode from HOLD
W#16#513E/503E	Wait point 3 "mode change from STARTUP to RUN" reached/passed
W#16#5140	Start of cycle
W#16#5142	Start process image updating
W#16#5144	Start user program processing
W#16#5146	RUN reached after mode transition HOLD-RUN
W#16#5148	OB start
W#16#5150	Start of HOLD loop
W#16#515E/505E	Wait point 4/5 "exit HOLD mode" reached/passed
W#16#515F	End of remaining cycle after restart
W#16#5961	Parameter assignment error
W#16#5962	Parameter assignment error preventing startup
W#16#5963	Parameter assignment error with memory reset request
W#16#5164	Enable SDB1 evaluation
W#16#5165/5065	Consistency error preventing startup entering/leaving state
W#16#516E	Partial parameter assignment requested
W#16#516F	Complete parameter assignment requested
W#16#5170	Distributed I/Os: evaluation of configuration information started

Event ID	Event
W#16#5371	Distributed I/Os: end of the synchronization with a DP master
W#16#5172	Distributed I/Os: deactivate
W#16#5175	Distributed I/Os: station return
W#16#5176	Distributed I/Os: reassignment of parameters to the station
W#16#5177	Distributed I/Os: complete parameter assignment during startup
W#16#5178	Distributed I/Os: insertion of a module
W#16#5191	Distributed I/Os: start of synchronization with a DP master
W#16#5192	Distributed I/Os: error in synchronization with a DP master
W#16#5193	Distributed I/Os: startup disabled during synchronization with a DP master
W#16#5395	Distributed I/Os: reset of a DP master
W#16#5196/5096	Distributed I/Os: system connection to a DP master failed/ok
W#16#5197	Distributed I/Os: event during startup synchronization

## C.7 Event Class 6 – Communication Events

Event ID	Event
W#16#6500	Connection ID exists twice on module
W#16#6501	Connection resources inadequate
W#16#6502	Error in the connection description
W#16#6503	Incorrect parameter assignment for MPI connection
W#16#6510	CFB structure error detected in instance DB when evaluating EPROM
W#16#6514	GD packet number exists twice on the module
W#16#6515	Inconsistent length specifications in GD configuration information
W#16#6316	Interface error when starting programmable controller
W#16#6521	No memory submodule and no internal memory available
W#16#6522	Illegal memory submodule: replace submodule and reset memory
W#16#6523	Memory reset request due to error accessing submodule
W#16#6524	Memory reset request due to error in block header
W#16#6526	Memory reset request due to memory replacement
W#16#6527	Memory replaced, therefore restart not possible
W#16#6528	Object handling function in the STOP/HOLD mode, no restart possible
W#16#6529	No startup possible during the "load user program" function
W#16#652A	No startup because block exists twice in user memory
W#16#652B	No startup because block is too long for submodule – replace submodule
W#16#652C	No startup due to illegal OB on submodule
W#16#6031/6131	Compress start/end
W#16#6532	No startup because illegal configuration information on submodule
W#16#6533	Memory reset request because of invalid submodule content
W#16#6534	No startup: block exists more than once on submodule
W#16#6535	No startup: not enough memory to transfer block from submodule
W#16#6536	No startup: submodule contains an illegal block number
W#16#6537	No startup: submodule contains a block with an illegal length
W#16#6538	Local data or write-protection ID (for DB) of a block illegal for CPU
W#16#6539	Illegal command in block (detected by compiler)
W#16#653A	Memory reset request because local OB data on submodule too short

Event ID	Event
W#16#6340	User program transfer from load memory to work memory start check, start of user program
W#16#6240	User program transfer from load memory to work memory start check, end of user program
W#16#6141/6041	Startup disabled due to DP interface module start/end
W#16#6142/6042	Startup disabled due to loading user program start/end
W#16#6543	No startup: illegal block type
W#16#6544	No startup: attribute "relevant for processing" illegal
W#16#6545	Source language illegal
W#16#6546	Maximum amount of configuration information reached
W#16#6547	Parameter assignment error assigning parameters to modules (not on P bus, cancel download)
W#16#6548	Plausibility error during block check
W#16#6560	PMC: SCAN overflow



## C.8 Event Class 8 – Diagnostic Events for Modules

Event ID	Event	Module type
W#16#8x00	Module: fault/ok	Any
W#16#8x01	Internal error	
W#16#8x02	External error	
W#16#8x03	Channel error	
W#16#8x04	No external auxiliary voltage	
W#16#8x05	No front connector	
W#16#8x06	No parameter assignment	
W#16#8x07	Incorrect parameters in module	
W#16#8x30	User submodule incorrect/not found	
W#16#8x31	Communication: problem	
W#16#8x32	Operating mode: RUN/STOP (STOP: entering state, RUN: leaving state)	
W#16#8x33	Watchdog responded:	
W#16#8x34	Internal module power: failure	
W#16#8x35	BATTF: battery: exhausted	
W#16#8x36	Total backup: failed	
W#16#8x37	Reserved	
W#16#8x40	Expansion rack: failed	
W#16#8x41	Processor failure:	
W#16#8x42	EPROM error:	
W#16#8x43	RAM error:	
W#16#8x44	ADC/DAC error:	
W#16#8x45	Fuse blown:	
W#16#8x46	Hardware interrupt lost:	
W#16#8x47	Reserved	
W#16#8x50	configuration/parameter assignment error:	Analog input
W#16#8x51	common mode error:	
W#16#8x52	short circuit to phase	
W#16#8x53	short circuit to ground	
W#16#8x54	wire break/supply current monitoring: MU/Pt100	
W#16#8x55	reference channel error	
W#16#8x56	below measuring range (3 mA)	
W#16#8x57	above measuring range (> 22 mA)	

Event ID	Event	Module type
W#16#8x60	configuration/parameter assignment error:	Analog output
W#16#8x61	common mode error:	
W#16#8x62	short circuit to phase:	
W#16#8x63	short circuit to ground:	
W#16#8x64	wire break/supply current monitoring: MU/Pt100	
W#16#8x65	Reserved	
W#16#8x66	No load voltage:	
W#16#8x70	configuration/parameter assignment error:	Digital input
W#16#8x71	chassis ground fault:	
W#16#8x72	short circuit to phase (sensor):	
W#16#8x73	short circuit to ground (sensor):	
W#16#8x74	wire break:	
W#16#8x75	no sensor power supply:	
W#16#8x80	configuration/parameter assignment error:	Digital output
W#16#8x81	chassis ground fault:	
W#16#8x82	short circuit to phase:	
W#16#8x83	short circuit to ground:	
W#16#8x84	wire break:	
W#16#8x85	fused tripped:	
W#16#8x86	No load voltage:	
W#16#8x87	overtemperature:	
W#16#8xB0	Counter module, signal A faulty	FM
W#16#8xB1	Counter module, signal B faulty	
W#16#8xB2	Counter module, signal N faulty	
W#16#8xB3	Counter module, incorrect value passed between the channels	
W#16#8xB4	Counter module, 5.2 V sensor supply faulty	
W#16#8xB5	Counter module, 24 V sensor supply faulty	

## C.9 Event Class 9 – Standard User Events

Event ID	Event
W#16#9001	Manual (1)/automatic (0) mode: status:
W#16#9101	Manual (1)/automatic (0) mode: status:
W#16#9x02	OPEN/CLOSED, ON/OFF
W#16#9x03	Manual command enable
W#16#9x04	Unit protective command (OPEN/CLOSED)
W#16#9x05	Process enable:
W#16#9x06	System protection command:
W#16#9x07	Process value monitoring responded:
W#16#9x08	Manipulated variable monitoring responded:
W#16#9x09	System deviation greater than permitted:
W#16#9x0A	Limit position error:
W#16#9x0B	Runtime error:
W#16#9x0C	Command execution error (sequencer):
W#16#9x0D	Operating status running > OPEN:
W#16#9x0E	Operating status running > CLOSED:
W#16#9x0F	Command blocking:
W#16#9x11	Process status OPEN/ON:
W#16#9x12	Process status CLOSED/OFF:
W#16#9x13	Process status intermediate position:
W#16#9x14	Process status ON via AUTO:
W#16#9x15	Process status ON via manual:
W#16#9x16	Process status ON via protective command:
W#16#9x17	Process status OFF via AUTO:
W#16#9x18	Process status OFF via manual:
W#16#9x19	Process status OFF via protective command:
W#16#9x21	Function error on approach:
W#16#9x22	Function error on leaving:
W#16#9x31	Actuator (DE/WE) limit position OPEN:
W#16#9x32	Actuator (DE/WE) limit position not OPEN:
W#16#9x33	Actuator (DE/WE) limit position CLOSED:
W#16#9x34	Actuator (DE/WE) limit position not CLOSED:
W#16#9x41	Illegal status, tolerance time elapsed:
W#16#9x42	Illegal status, tolerance time not elapsed:
W#16#9x43	Interlock error, tolerance time = 0:

Event ID	Event
W#16#9x44	Interlock error, tolerance time > 0:
W#16#9x45	No reaction:
W#16#9x46	Final status exited illegally, tolerance time = 0:
W#16#9x47	Final status exited illegally, tolerance time > 0:
W#16#9x50	Upper limit of signal range USR:
W#16#9x51	Upper limit of measuring range UMR:
W#16#9x52	Lower limit of signal range LSR:
W#16#9x53	Lower limit of measuring range LMR:
W#16#9x54	Upper alarm limit UAL:
W#16#9x55	Upper warning limit UWL:
W#16#9x56	Upper tolerance limit UTL:
W#16#9x57	Lower tolerance limit LTL:
W#16#9x58	Lower warning limit LWL:
W#16#9x59	Lower alarm limit LAL:
W#16#9x60	GRAPH7 step entering/leaving:
W#16#9x61	GRAPH7 interlock error:
W#16#9x62	GRAPH7 execution error:
W#16#9x63	GRAPH7 error noted:
W#16#9x64	GRAPH7 error acknowledged:
W#16#9x70	Trend exceeded in positive direction:
W#16#9x71	Trend exceeded in negative direction:
W#16#9x72	No reaction:
W#16#9x73	Final state exited illegally:
W#16#9x80	Limit value exceeded, tolerance time = 0:
W#16#9x81	Limit value exceeded, tolerance time > 0:
W#16#9x82	Below limit value, tolerance time = 0:
W#16#9x83	Below limit value, tolerance time > 0:
W#16#9x84	Gradient exceeded, tolerance time = 0:
W#16#9x85	Gradient exceeded, tolerance time > 0:
W#16#9x86	Below gradient, tolerance time = 0:
W#16#9x87	Below gradient, tolerance time > 0:
W#16#9190/9090	User parameter assignment error:
W#16#91F0	Overflow:
W#16#91F1	Underflow:
W#16#91F2	Division by 0:
W#16#91F3	Illegal calculation operation:

## C.10 Event Classes A and B – Free User Events

Event ID	Event
W#16#Axyz	Events available for user
W#16#Bxyz	

## C.11 Reserved Event Classes

### Reserved

The following event classes are reserved for later expansions:

- C
- D
- E
- F

Reserved for modules not in central rack (for example CPs or FMs)

## List of SFCs, SFBs and FCs

# D

Section	Description	Page
D.1	List of SFCs, Sorted Numerically	D-2
D.2	List of SFCs, Sorted Alphabetically	D-4
D.3	List of SFBs, Sorted Numerically	D-6
D.4	List of SFBs, Sorted Alphabetically	D-7
D.5	List of FCs	D-8

## D.1 List of SFCs, Sorted Numerically

### System Functions, Sorted Numerically

Table D-1 List of all SFCs, Sorted Numerically

No.	Short Name	Function	Section
SFC0	SET_CLK	Set System Clock	5.1
SFC1	READ_CLK	Read System Clock	5.2
SFC2	SET_RTM	Set Run-Time Meter	6.2
SFC3	CTRL_RTM	Start/Stop Run-Time Meter	6.3
SFC4	READ_RTM	Read Run-Time Meter	6.4
SFC5	GADR_LGC	Query Logical Address of a Channel	14.1
SFC6	RD_SINFO	Read OB Start Information	12.1
SFC9	EN_MSG	Enable Block-Related, Symbol-Related and Group Status Messages	19.8
SFC10	DIS_MSG	Disable Block-Related, Symbol-Related and Group Status Messages	19.7
SFC13	DPNRM_DG	Read Diagnostic Data of a DP Slave (Slave Diagnostics)	15.2
SFC14	DPRD_DAT	Read Consistent Data of a Standard DP Slave	15.3
SFC15	DPWR_DAT	Write Consistent Data to a DP Standard Slave	15.4
SFC17	ALARM_SQ	Generate Acknowledgable Block-Related Messages	19.12
SFC18	ALARM_S	Generate Permanently Acknowledged Block-Related Messages	19.12
SFC19	ALARM_SC	Query the Acknowledgment Status of the last ALARM_SQ Entering State Message	19.13
SFC20	BLKMOV	Copy Variables	3.1
SFC21	FILL	Initialize a Memory Area	3.2
SFC22	CREAT_DB	Create Data Block	3.3
SFC23	DEL_DB	Delete Data Block	3.4
SFC24	TEST_DB	Test Data Block	3.5
SFC25	COMPRESS	Compress the User Memory	3.6
SFC26	UPDAT_PI	Update the Process Image Update Table	13.1
SFC27	UPDAT_PO	Update the Process Image Output Table	13.2
SFC28	SET_TINT	Set Time-of-Day Interrupt	8.3
SFC29	CAN_TINT	Cancel Time-of-Day Interrupt	8.4
SFC30	ACT_TINT	Activate Time-of-Day Interrupt	8.5
SFC31	QRY_TINT	Query Time-of-Day Interrupt	8.6
SFC32	SRT_DINT	Start Time-Delay Interrupt	9.2
SFC33	CAN_DINT	Cancel Time-Delay Interrupt	9.4
SFC34	QRY_DINT	Query Time-Delay Interrupt	9.3
SFC35	MP_ALM	Trigger Multicomputing Interrupt	4.4
SFC36	MSK_FLT	Mask Synchronous Errors	10.2
SFC37	DMSK_FLT	Unmask Synchronous Errors	10.3
SFC38	READ_ERR	Read Error Register	10.4
SFC39	DIS_IRT	Disable New Interrupts and Asynchronous Errors	11.2
SFC40	EN_IRT	Enable New Interrupts and Asynchronous Errors	11.3



Table D-1 List of all SFCs, Sorted Numerically, continued

No.	Short Name	Function	Section
SFC41	DIS_AIRT	Delay Higher Priority Interrupts and Asynchronous Errors	11.4
SFC42	EN_AIRT	Enable Higher Priority Interrupts and Asynchronous Errors	11.5
SFC43	RE_TRIGR	Retrigger Cycle Time Monitoring	4.1
SFC44	REPL_VAL	Transfer Substitute Value to Accumulator 1	3.7
SFC46	STP	Change the CPU to STOP	4.2
SFC47	WAIT	Delay Execution of the User Program	4.3
SFC48	SNC_RTCB	Synchronize Slave Clocks	5.3
SFC49	LGC_GADR	Query the Module Slot Belonging to a Logical Address	14.2
SFC50	RD_LGADR	Query all Logical Addresses of a Module	14.3
SFC51	RDSYSST	Read a System Status List or Partial List	12.2
SFC52	WR_USMSG	Write a User-Defined Diagnostic Event to the Diagnostic Buffer	12.3
SFC55	WR_PARM	Write Dynamic Parameters	7.2
SFC56	WR_DPARM	Write Default Parameters	7.3
SFC57	PARM_MOD	Assign Parameters to a Module	7.4
SFC58	WR_REC	Write a Data Record	7.5
SFC59	RD_REC	Read a Data Record	7.6
SFC60	GD_SND	Send a GD Packet	16.1
SFC61	GD_RCV	Fetch a Received GD Packet	16.2
SFC62	CONTROL	Query the Status of a Connection Belonging to a Communication SFB Instance	17.15
SFC63	AB_CALL	Assembly Code Block	*
SFC64	TIME_TCK	Read the System Time	6.5
SFC65	X_SEND	Send Data to a Communication Partner outside the Local S7 Station	18.5
SFC66	X_RCV	Receive Data from a Communication Partner outside the Local S7 Station	18.6
SFC67	X_GET	Read Data from a Communication Partner outside the Local S7 Station	18.7
SFC68	X_PUT	Write Data to a Communication Partner outside the Local S7 Station	18.8
SFC69	X_ABORT	Abort an Existing Connection to a Communication Partner outside the Local S7 Station	18.9
SFC72	I_GET	Read Data from a Communication Partner within the Local S7 Station	18.10
SFC73	I_PUT	Write Data to a Communication Partner within the Local S7 Station	18.11
SFC74	I_ABORT	Abort an Existing Connection to a Communication Partner within the Local S7 Station	18.12
SFC79	SET	Set a Range of Outputs	13.3
SFC80	RSET	Reset a Range of Outputs	13.4

\* SFC63 “AB\_CALL” only exists for CPU 614. For a detailed description, refer to the corresponding manual.

## D.2 List of SFCs, Sorted Alphabetically

Table D-2 List of all SFCs, Sorted Alphabetically

Short Name	No.	Function	Section
AB_CALL	SFC63	Assembly Code Block	*
ACT_TINT	SFC30	Activate Time-of-Day Interrupt	8.5
ALARM_SQ	SFC17	Generate Acknowledgable Block-Related Messages	19.12
ALARM_S	SFC18	Generate Permanently Acknowledged Block-Related Messages	19.12
ALARM_SC	SFC19	Query the Acknowledgment Status of the last ALARM_SQ Entering State Message	19.13
BLKMOV	SFC20	Copy Variables	3.1
CAN_DINT	SFC33	Cancel Time-Delay Interrupt	9.4
CAN_TINT	SFC29	Cancel Time-of-Day Interrupt	8.4
COMPRESS	SFC25	Compress the User Memory	3.6
CONTROL	SFC62	Query the Status of a Connection Belonging to a Communication SFB Instance	17.15
CREAT_DB	SFC22	Create Data Block	3.3
CTRL_RTM	SFC3	Start/Stop Run-Time Meter	6.3
DEL_DB	SFC23	Delete Data Block	3.4
DIS_AIRT	SFC41	Delay the Higher Priority Interrupts and Asynchronous Errors	11.4
DIS_IRT	SFC39	Disable New Interrupts and Asynchronous Errors	11.2
DIS_MSG	SFC10	Disable Block-Related, Symbol-Related and Group Status Messages	19.7
DMSK_FLT	SFC37	Unmask Synchronous Errors	10.3
DPNRM_DG	SFC13	Read Diagnostic Data of a DP Slave (Slave Diagnostics)	15.2
DPRD_DAT	SFC14	Read Consistent Data of a Standard DP Slave	15.3
DPWR_DAT	SFC15	Write Consistent Data to a DP Standard Slave	15.4
EN_AIRT	SFC42	Enable Higher Priority Interrupts and Asynchronous Errors	11.5
EN_IRT	SFC40	Enable New Interrupts and Asynchronous Errors	11.3
EN_MSG	SFC9	Enable Block-Related, Symbol-Related and Group Status Messages	19.8
FILL	SFC21	Initialize a Memory Area	3.2
GADR_LGC	SFC5	Query Logical Address of a Channel	14.1
GD_RCV	SFC61	Fetch a Received GD Packet	16.2
GD_SND	SFC60	Send a GD Packet	16.1
I_ABORT	SFC74	Abort an Existing Connection to a Communication Partner within the Local S7 Station	18.12
I_GET	SFC72	Read Data from a Communication Partner within the Local S7 Station	18.10
I_PUT	SFC73	Write Data to a Communication Partner within the Local S7 Station	18.11
LGC_GADR	SFC49	Query the Module Slot Belonging to a Logical Address	14.2
MP_ALM	SFC35	Trigger Multicomputing Interrupt	4.4
MSK_FLT	SFC36	Mask Synchronous Errors	10.2
PARAM_MOD	SFC57	Assign Parameters to a Module	7.4
QRY_DINT	SFC34	Query Time-Delay Interrupt	9.3
QRY_TINT	SFC31	Query Time-of-Day Interrupt	8.6

Table D-2 List of all SFCs, Sorted Alphabetically, continued

Short Name	No.	Function	Section
RD_LGADR	SFC50	Query all Logical Addresses of a Module	14.3
RD_REC	SFC59	Read a Data Record	7.6
RD_SINFO	SFC6	Read OB Start Information	12.1
RDSYSST	SFC51	Read a System Status List or Partial List	12.2
READ_CLK	SFC1	Read System Clock	5.2
READ_RTM	SFC4	Read Run-Time Meter	6.4
READ_ERR	SFC38	Read Error Registers	10.4
REPL_VAL	SFC44	Transfer Substitute Value to Accumulator 1	3.7
RE_TRIGR	SFC43	Retrigger Cycle Time Monitoring	4.1
RSET	SFC80	Reset a Range of Outputs	13.4
SET	SFC79	Set a Range of Outputs	13.3
SET_CLK	SFC0	Set System Clock	5.1
SET_RTM	SFC2	Set Run-Time Meter	6.2
SET_TINT	SFC28	Set Time-of-Day Interrupt	8.3
SNC_RTCB	SFC48	Synchronize Slave Clocks	5.3
SRT_DINT	SFC32	Start Time-Delay Interrupt	9.2
STP	SFC46	Change the CPU to STOP	4.2
TEST_DB	SFC24	Test Data Block	3.5
TIME_TCK	SFC64	Read the System Time	6.5
UPDAT_PI	SFC26	Update the Process Image Update Table	13.1
UPDAT_PO	SFC27	Update the Process Image Output Table	13.2
WAIT	SFC47	Delay Execution of the User Program	4.3
WR_DPARM	SFC56	Write Default Parameters	7.3
WR_PARM	SFC55	Write Dynamic Parameters	7.2
WR_REC	SFC58	Write data Record	7.5
WR_USMSG	SFC52	Write a User-Defined Diagnostic Event to the Diagnostic Buffer	12.3
X_ABORT	SFC69	Abort an Existing Connection to a Communication Partner outside the Local S7 Station	18.9
X_GET	SFC67	Read Data from a Communication Partner outside the Local S7 Station	18.7
X_PUT	SFC68	Write Data to a Communication Partner outside the Local S7 Station	18.8
X_RCV	SFC66	Receive Data from a Communication Partner outside the Local S7 Station	18.6
X_SEND	SFC65	Send Data to a Communication Partner outside the Local S7 Station	18.5

\* SFC63 "AB\_CALL" only exists for CPU 614. For a detailed description, refer to the corresponding manual.

### D.3 List of SFBs, Sorted Numerically

#### SFBs, Sorted Numerically

Table D-3 List of all SFBs, Sorted Numerically

No.	Short Name	Function	Section
SFB0	CTU	Count Up	20.4
SFB1	CTD	Count Down	20.5
SFB2	CTUD	Count Up/Down	20.6
SFB3	TP	Generate a Pulse	20.1
SFB4	TON	Generate an On Delay	20.2
SFB5	TOF	Generate an Off Delay	20.3
SFB8	USEND	Uncoordinated Sending of Data	17.3
SFB9	URCV	Uncoordinated Receiving of Data	17.4
SFB12	BSEND	Sending Segmented Data	17.5
SFB13	BRCV	Receiving Segmented Data	17.6
SFB14	GET	Read Data from a Remote CPU	17.7
SFB15	PUT	Write Data to a Remote CPU	17.8
SFB16	PRINT	Send Data to Printer	17.9
SFB19	START	Initiate a Complete Restart on a Remote Device	17.10
SFB20	STOP	Changing a Remote Device to the STOP State	17.11
SFB21	RESUME	Initiate a Restart on a Remote Device	17.12
SFB22	STATUS	Query the Status of a Remote Partner	17.13
SFB23	USTATUS	Receive the Status of a Remote Device	17.14
SFB29	HS_COUNT	Counter (high-speed counter, integrated function)	*
SFB30	FREQ_MES	Frequency Meter (frequency meter, integrated function)	*
SFB32	DRUM	Implement a Sequencer	13.5
SFB33	ALARM	Generate Block-Related Messages with Acknowledgment Display	19.3
SFB34	ALARM_8	Generate Block-Related Messages without Values for 8 Signals	19.5
SFB35	ALARM_8P	Generate Block-Related Messages with Values for 8 Signals	19.4
SFB36	NOTIFY	Generate Block-Related Messages without Acknowledgment Display	19.2
SFB37	AR_SEND	Send Archive Data	19.6
SFB38	HSC_A_B	Counter A/B (integrated function)	*
SFB39	POS	Position (integrated function)	*
SFB41 <sup>1)</sup>	CONT_C	Continuous Control	22.1
SFB42 <sup>1)</sup>	CONT_S	Step Control	22.2
SFB43 <sup>1)</sup>	PULSEGEN	Pulse Generation	22.3

\* SFB29 "HS\_COUNT" and SFB30 "FREQ\_MES" only exist on the CPU 312 IFM and CPU 314 IFM. SFBs 38 "HSC\_A\_B" and 39 "POS" only exist on the CPU 314 IFM. For a detailed description, refer to **/73/**.

<sup>1)</sup> SFBs 41 "CONT\_C", 42 "CONT\_S" and 43 "PULSEGEN" only exist on the CPU 314 IFM.

## D.4 List of SFBs, Sorted Alphabetically

Table D-4 List of all SFBs, Sorted Alphabetically

Short Name	No.	Function	Section
ALARM	SFB33	Generate Block-Related Messages with Acknowledgment	19.3
ALARM_8	SFB34	Generate Block-Related Messages without Values for 8 Signals	19.5
ALARM_8P	SFB35	Generate Block-Related Messages with Values for 8 Signals	19.4
AR_SEND	SFB37	Send Archive Data	19.6
BRCV	SFB13	Receiving Segmented Data	17.6
BSEND	SFB12	Sending Segmented Data	17.5
CONT_C <sup>1)</sup>	SFB41	Continuous Control	22.1
CONT_S <sup>1)</sup>	SFB42	Step Control	22.2
CTD	SFB1	Count Down	20.5
CTU	SFB0	Count Up	20.4
CTUD	SFB2	Count Up/Down	20.6
DRUM	SFB32	Implement a Sequencer	13.5
FREQ_MES	SFB30	Frequency Meter (frequency meter, integrated function)	*
GET	SFB14	Read Data from a Remote CPU	17.7
HSC_A_B	SFB38	Counter A/B (integrated function)	*
HS_COUNT	SFB29	Counter (high-speed counter, integrated function)	*
NOTIFY	SFB36	Generate Block-Related Messages without Acknowledgment Display	19.2
POS	SFB39	Position (integrated function)	*
PRINT	SFB16	Send Data to Printer	17.9
PULSEGEN <sup>1)</sup>	SFB43	Pulse Generation	22.3
PUT	SFB15	Write Data to a Remote CPU	17.8
RESUME	SFB21	Initiate a Restart on a Remote Device	17.12
START	SFB19	Initiate a Complete Restart on a Remote Device	17.10
STATUS	SFB22	Query the Status of a Remote Partner	17.13
STOP	SFB20	Changing a Remote Device to the STOP State	17.11
TOF	SFB5	Generate an Off Delay	20.3
TON	SFB4	Generate an On Delay	20.2
TP	SFB3	Generate a Pulse	20.1
URCV	SFB9	Uncoordinated Receiving of Data	17.4
USEND	SFB8	Uncoordinated Sending of Data	17.3
USTATUS	SFB23	Receive the Status of a Remote Device	17.14

\* SFB29 "HS\_COUNT" and SFB30 "FREQ\_MES" only exist for CPU 312 IFM and CPU 314 IFM. SFBs 38 "HSC\_A\_B" and 39 "POS" only exist on the CPU 314 IFM. For a detailed description, refer to **/73/**.

1) SFBs 41 "CONT\_C", 42 "CONT\_S" and 43 "PULSEGEN" only exist on the CPU 314 IFM.

## D.5 List of FCs

### IEC Functions, Sorted Numerically

Table D-5 List of all IEC Functions

No.	Short Name	Function	Section
FC1	AD_DT_TM	Add duration to a time	21.4
FC2	CONCAT	Combine two STRING variables	21.7
FC3	D_TOD_DT	Combine DATE and TIME_OF_DAY to DT	21.4
FC4	DELETE	Delete in a STRING variable	21.7
FC5	DI_STRNG	Data type conversion DINT to STRING	21.8
FC6	DT_DATE	Extract the DATE from DT	21.4
FC7	DT_DAY	Extract the day of the week from DT	21.4
FC8	DT_TOD	Extract the TIME_OF_DAY from DT	21.4
FC9	EQ_DT	Compare DT for equal	21.5
FC10	EQ_STRNG	Compare STRING for equal	21.6
FC11	FIND	Find in a STRING variable	21.7
FC12	GE_DT	Compare DT for greater than or equal	21.5
FC13	GE_STRNG	Compare STRING for greater than or equal	21.6
FC14	GT_DT	Compare DT for greater than	21.5
FC15	GT_STRNG	Compare STRING for greater than	21.6
FC16	I_STRNG	Data type conversion INT to STRING	21.8
FC17	INSERT	Insert in a STRING variable	21.7
FC18	LE_DT	Compare DT for less than or equal	21.5
FC19	LE_STRNG	Compare STRING for less than or equal	21.6
FC20	LEFT	Left part of a STRING variable	21.7
FC21	LEN	Length of a STRING variable	21.7
FC22	LIMIT	Limit	21.9
FC23	LT_DT	Compare DT for less than	21.5
FC24	LT_STRNG	Compare STRING for less than	21.6
FC25	MAX	Select maximum	21.9
FC26	MID	Middle part of a STRING variable	21.7
FC27	MIN	Select minimum	21.9
FC28	NE_DT	Compare DT for unequal	21.5
FC29	NE_STRNG	Compare STRING for unequal	21.6
FC30	R_STRNG	Data type conversion REAL to STRING	21.8
FC31	REPLACE	Replace in a STRING variable	21.7
FC32	RIGHT	Right part of a STRING variable	21.7
FC33	S5TI_TIM	Data type conversion S5TIME to TIME	21.4
FC34	SB_DT_DT	Subtract two time values	21.4
FC35	SB_DT_TM	Subtract duration from a time	21.4
FC36	SEL	Binary selection	21.10
FC37	STRNG_DI	Data type conversion STRING to DINT	21.8
FC38	STRNG_I	Data type conversion STRING to INT	21.8
FC39	STRNG_R	Data type conversion STRING to REAL	21.8
FC40	TIM_S5TI	Data type conversion TIME to S5TIME	21.4

# System Data Blocks (SDBs)

# E

## Chapter Overview

Section	Description	Page
E.1	System Data Blocks (SDBs)	E-2

## E.1 System Data Blocks (SDBs)

Table E-1 Overview of the SDBs used with S7

SDB No.	Content
0	Operating system parameters
1	Expected configuration of the PLC, assignment of logical to geographical (physical) addresses
2	Default operating system parameters
3	Submodules
4	Consistency SDB for multicomputing
5	List of party line subscribers
22 to 29	Lists for address conversion in DP
100 to 121	Parameters for S7 modules that are not located in the DP area
122 to 129	Parameters for S7 modules in the DP area
150 to 179	Parameters for submodules
210	Global data
300 to 328	Configured data for individual SCAN cycles
701 to 999	Description of configured homogeneous connections
1000 to 32767	Parameter data records for communication (K) bus, DP master and DP slave modules
33792 to 34815	Reserved for TDs/OPs



# References

# F

- /30/ Primer: *S7-300 Programmable Controller, Quick Start*
- /70/ Manual: *S7-300 Programmable Controller, Hardware and Installation*
- /71/ Reference Manual: *S7-300, M7-300 Programmable Controllers Module Specifications*
- /72/ Instruction List: *S7-300 Programmable Controller*
- /73/ Manual: *S7-300 Programmable Controller, Integrated Functions CPU 312 IFM/314 IFM*
- /100/ Manual: *S7-400/M7-400 Programmable Controllers, Hardware and Installation*
- /101/ Reference Manual: *S7-400/M7-400 Programmable Controllers Module Specifications*
- /102/ Instruction List: *S7-400 Programmable Controller*
- /230/ Manual: *Standard Software for S7, Converting S5 Programs*
- /231/ User Manual: *Standard Software for S7 and M7, STEP 7*
- /232/ Manual: *Statement List (STL) for S7-300 and S7-400, Programming*
- /233/ Manual: *Ladder Logic (LAD) for S7-300 and S7-400, Programming*
- /234/ Programming Manual: *System Software for S7-300 and S7-400 Program Design*
- /236/ Manual: *Function Block Diagram (FBD) for S7-300 and S7-400, Programming*
- /250/ Manual: *Structured Control Language (SCL) for S7-300 and S7-400, Programming*
- /251/ Manual: *GRAPH for S7-300 and S7-400, Programming Sequential Control Systems*
- /252/ Manual: *HiGraph for S7-300 and S7-400, Programming State Graphs*

- /254/ Manual: *Continuous Function Charts (CFC) for S7 and M7*,  
Programming Continuous Function Charts
- /270/ Manual: *S7-PDIAG for S7-300 and S7-400*,  
“Configuring Process Diagnostics for LAD, STL, and FBD”
- /350/ User Manual: *SIMATIC 7*,  
*Standard Control*

# Glossary

## A

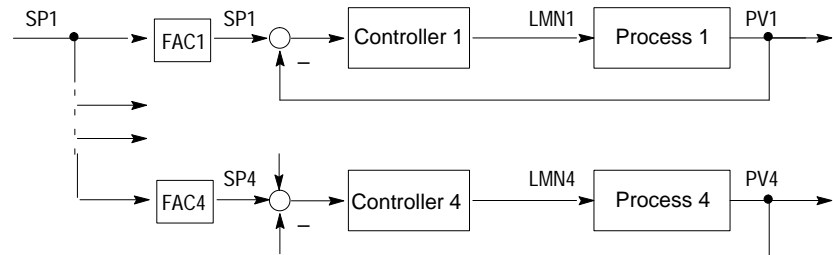
<b>Address</b>	The address is the identifier given to a memory location or range of memory locations, for example: input I 12.1; bit memory MW25; data block DB3.
<b>Addressing</b>	Assigning an address in the user program. Addresses can be assigned to a memory location or range of memory locations, for example: input I 12.1; bit memory MW25; data block DB3.
<b>ACCU (Accumulator)</b>	Accumulators are registers in the → CPU and serve as buffers for load and transfer operations, as well as for comparison, math and conversion operations.
<b>Actual Parameter</b>	Actual parameters replace formal parameters when a function block (FB) or function (FC) is called, for example, the formal parameter “REQ” is replaced by the actual parameter “I 3.6”.

## B

<b>Bit Memory</b>	This is a 1 bit memory location. Bit memory allows write and read access with STEP 7 basic operations (addressing using bits, bytes, words and double words). The user can use the bit memory address area to save interim results.
-------------------	---

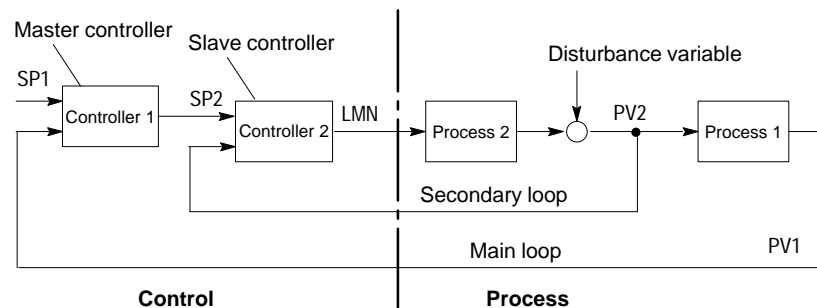
**Blending control**

Blending control involves a controller structure in which the setpoint for the total amount SP is converted to percentages of the individual components. The total of the blending factors FAC must be 1 (= 100 %).

**C****Cascade control**

Cascade control involves a series of interconnected controllers, in which the master controller adjusts the setpoint for the secondary (slave) controllers according to the instantaneous error signal of the main process variable.

A cascade control system can be improved by including additional process variables. A secondary process variable PV2 is measured at a suitable point and controlled to the reference setpoint (output of the master controller SP2). The master controller controls the process variable PV1 to the fixed setpoint SP1 and sets SP2 so that the target is achieved as quickly as possible without overshoot.

**Closed-loop controller**

A closed-loop controller is a device in which the error signal is continuously calculated and an actuating signal generated with the aim of eliminating the error signal quickly and without overshoot.

**Communication, bilateral**

When using → communication SFBs for data exchange, a distinction is made between unilateral and bilateral communication. Communication is bilateral when there is a SFB on the local and the remote module, for example the communication SFBs "USEND" and "URCV".

<b>Communication SFBs for configured connections</b>	<p>Communication SFBs are system function blocks for data exchange and program management.</p> <p>Examples of data exchange: SEND, RECEIVE, GET.</p> <p>Examples of program management: setting the CPU of a communication partner to the STOP state, querying the STATUS of the CPU of a communication partner.</p>
<b>Communication SFCs for non-configured Connections</b>	<p>Communication SFCs are system functions for data exchange and for aborting existing connections established by the communication SFCs.</p>
<b>Communication, unilateral</b>	<p>When using → communication SFBs for data exchange, a distinction is made between unilateral and bilateral communication. Communication is unilateral when there is a SFB only on the local module, for example the SFB “GET”.</p>
<b>Complete restart</b>	<p>When a CPU starts up (for example, when the mode selector is moved from STOP to RUN or when power is turned on), before cyclic program processing starts (OB1), either the organization block OB101 (restart; only in the S7-400) or OB100 (complete restart) is processed first. In a complete restart the process-image input table is read in and the STEP 7 user program processed starting with the first statement in OB1.</p>
<b>Constant</b>	<p>“Constants” are token values for constant values in → logic blocks. Constants are used to improve the legibility of a program. For example, instead of entering a value directly (for example, 10), the token value “Max_iteration_count” is entered in a → function block. The value of the constant (for example, 10) is then entered when the block is called.</p>
<b>Continuous controller</b>	<p>A continuous controller is a controller in which every change in the error signal produces a change in the manipulated variable. This can adopt any value within the range of the manipulated variable.</p>
<b>Control loop</b>	<p>The control loop is the connection between the process output (process variable) and the controller input and between the controller output (manipulated variable) and the process input, so that the controller and process form a closed loop.</p>
<b>Controller parameters</b>	<p>Controller parameters are characteristic values for the static and dynamic adaptation of the controller response to the given loop or process characteristics.</p>
<b>CPU Operating System</b>	<p>The CPU operating system organizes all functions and processes of the CPU that are not linked to a special control task.</p>

## D

<b>Data Block (DB)</b>	Data blocks are areas in the user program which contain user data. There are shared data blocks which can be accessed by all logic blocks, and there are instance data blocks which are associated with a particular function block (FB) call.
<b>Diagnostics</b>	Diagnostic functions incorporate all the system diagnostics and include the recognition, interpretation and reporting of errors within the PLC.
<b>Diagnostic Interrupt</b>	Diagnostic modules report recognized system errors using diagnostic interrupts to the → CPU.
<b>Diagnostic Data</b>	Diagnostic data is information contained in an error message (diagnostic event, time stamp).
<b>Diagnostic Entry</b>	A diagnostic event is described in the diagnostic buffer using a diagnostic entry.
<b>Diagnostic Message</b>	The diagnostic message consists of a processed diagnostic event and is sent from the CPU to the display unit.
<b>Diagnostic Buffer</b>	The diagnostic buffer is a memory area in the CPU in which all diagnostic events are stored in the order in which they occurred.

## E

<b>Error, asynchronous</b>	Asynchronous errors are → run time errors which are not assigned to any particular place in the user program (for example, power supply error, scan time overrun). When these errors occur, the operating system calls the corresponding → organization blocks in which the user can program a reaction.
<b>Error handling with OBs</b>	If the system program recognizes a particular error (for example, → access error in S7), it will call the designated organization block in which the CPU's response to the error can be set by the user program.

<b>Error OB</b>	Error OBs are organization blocks which the user can use to program the reaction to an error. However, a programmed reaction to an error is only possible if the error does not cause the PLC to stop. There is an error OB for each type of error. (For example, error OB for → addressing error, error OB for → access error in S7.)
<b>Error reaction</b>	Reaction to a → run-time error. The operating system can react in the following ways: by changing the PLC to the STOP status, by calling an organization block in which the user can program a reaction, or by displaying the error.
<b>Error, synchronous</b>	Synchronous errors are → run-time errors assigned to a particular place in the user program (for example, error accessing an I/O module). When these errors occur, the operating system calls the corresponding → organization blocks in which the user can program a reaction.
<b>Error, system error</b>	System errors are errors which may occur within a PLC (not in the process). System errors can be, for example → program errors in the CPU and faults in modules.

## F

<b>Formal parameter</b>	A formal parameter is a placeholder for the actual parameter in → logic blocks that can be assigned parameters. In FBs and FCs, the formal parameters are declared by the user; in SFBs and SFCs, they already exist. When a block is called, an actual parameter is assigned to the formal parameter so that the called block works with the latest value. The formal parameters belong to the → local data of the block and are declared as input, output, and in/out parameters.
-------------------------	---

## G

<b>Group error</b>	Error message indicated by a LED display on the front panel of modules (only) in S7-300. The LED lights up whenever there is an error in the module concerned ( → internal errors and → external errors).
--------------------	---

## H

**Hardware interrupt** A hardware interrupt is triggered by modules with interrupt capability as a result of a specific event in the process. The hardware interrupt is reported to the CPU. The assigned → organization block is then processed according to the priority of this interrupt.

## I

**Input parameter** Input parameters only exist in → functions and → function blocks. With the help of the input parameters, data are transferred to the called block for processing.

**Integral component** Integral component of the controller.  
After a step change in the process variable (or error signal) the output variable changes with a ramp function over time at a rate of change proportional to the integral-action factor  $KI (= 1/TI)$ . The integral component in a closed control loop has the effect of correcting the controller output variable until the error signal becomes zero.

**Integrated controller** An integrated controller is a ready programmed controller block available in the operating system and containing the most important functions of a closed-loop control application. The user can select and deselect functions using software switches.

**Interrupt** The → SIMATIC S7 priority class system recognizes 10 different priority classes, which regulate the processing of the user program. Interrupts belong to these priority classes, for example hardware interrupts. When an interrupt occurs, the operating system automatically calls an organization block in which the user can program the required reaction (for example, in a function block).

**Interrupt, time-of-day** The time-of-day interrupt belongs to one of the priority classes in SIMATIC S7 program execution. It is generated at a specific date (or day) and time (for example, 9:50 or every hour or every minute). A corresponding organization block is then executed.

**Interrupt, time-delay** The time-delay interrupt belongs to one of the priority classes in SIMATIC S7 program execution. It is generated when a timer has expired in the user program. A corresponding organization block is then executed.



## L

<b>Logic Block</b>	<p>In SIMATIC S7, a logic block is a block that contains part of the STEP 7 user program. The other type of block is a → data block which contains only data. The following list shows the types of logic blocks:</p> <ul style="list-style-type: none"><li>• Organization block (OB)</li><li>• Function block (FB)</li><li>• Function (FC)</li><li>• System function block (SFB)</li><li>• System function (SFC)</li></ul>
--------------------	---

## M

<b>Module Parameter</b>	<p>Module parameters are values with which the behavior of the module can be set. Depending on the particular module, some of these parameters can be modified in the user program.</p>
-------------------------	---

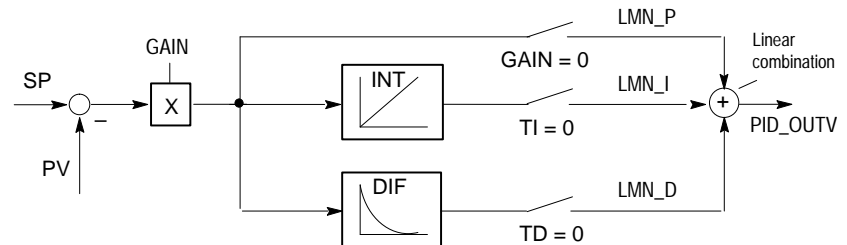
## O

<b>OB1</b>	<p>The organization block OB1 is the user interface for the system program for → cyclic program processing.</p>
<b>OB priority</b>	<p>The → operating system of the CPU differentiates between various priority classes, for example, cyclic program processing, hardware interrupt-controlled program processing. Organization blocks → (OB) are assigned to each priority class, in which the S7 user can program a reaction. The OBs have different priorities, which allow them to be processed in the correct sequence when two occur at the same time and allow OBs with higher priority to interrupt those with lower priority. The S7 user can change the standard priorities.</p>
<b>Organization block (OB)</b>	<p>Organization blocks form the interface between the CPU operating system and the user program. The sequence in which the user program is processed is specified in the organization blocks.</p>

## P

### Parallel structure

The parallel structure is a special type of signal processing in the controller (mathematical processing). The P, I and D components are calculated parallel to each other with no interaction and then totalled.



### Parameter

1. A parameter is a variable of an S7 logic block  
(see → block parameter → actual parameter, → formal parameter)
2. A variable for setting the behavior of a module  
(one or more per module)

Every configurable module has a basic parameter setting when it is supplied from the factory, but this can be changed using STEP 7.  
(one or more per module).

There are two types of parameter:

static and dynamic parameters (→ Parameter, static/ Parameter dynamic).

### Parameter, dynamic

Dynamic parameters of modules, in contrast to static parameters, can be changed by the user program during operation by calling an SFC, for example limit values of an analog module.

### Parameter, static

Static parameters of modules, in contrast to dynamic parameters, cannot be changed by the user program, but only using STEP 7, for example the input delay of a digital input module.

### P algorithm

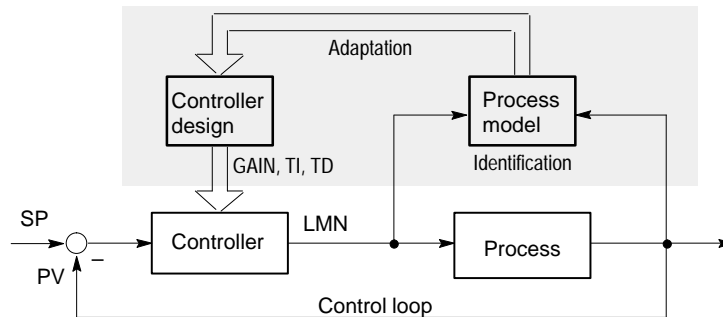
Algorithm for calculating an output signal in which there is a proportional relationship between the error signal and manipulated variable change.  
Characteristics: steady-state error signal, not to be used with processes including dead time.

### PI algorithm

Algorithm for calculating an output signal in which the change in the manipulated variable is made up of a component proportional to the error signal and an I component proportional to the error signal and time.  
Characteristics: no steady-state error signal, faster compensation than with an I algorithm, suitable for all processes.

**PID algorithm**

Algorithm for calculating an output signal formed by multiplication, integration and differentiation of the error signal. The PID algorithm is a → parallel structure. Characteristics: high degree of control quality can be achieved providing the dead time of the process is not greater than the other time constants.

**Priority**

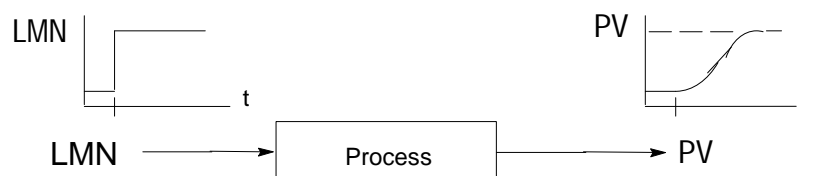
When you assign a priority to an organization block, you determine the interruptability of the currently active user program so that high-priority events interrupt lower-priority events.

**Priority class**

The operating system of a CPU has a maximum of 28 priority classes, to which the various organization blocks are assigned. The priority classes decide which OBs can interrupt other OBs. If a priority class includes more than one OB, these do not interrupt each other but are executed sequentially.

**Process**

The process is the part of the system in which the process variable is influenced by the manipulated variable (by changing the level of energy or mass). The process can be divided into the actuator and the actual process being controlled.

**Program execution, event-controlled**

With event-controlled program execution, the running of the cyclic user program is interrupted by start events (→ Priority classes). If a start event occurs, the block currently being executed is interrupted before the next instruction and an assigned organization block called and executed. Cyclic program execution then continues from the point of interruption.

**Proportional actuator**

→ Pulse duration modulation

## Pulse duration modulation

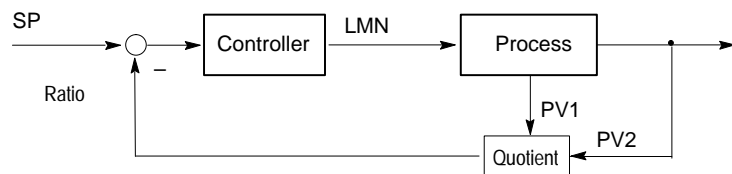
Pulse duration modulation is a method of influencing the manipulated variable at a discontinuous output. The calculated manipulated value as a percentage is converted to a proportional signal pulse time  $T_p$  at the manipulated variable output, for example,  $100\% T_p = T_A$  or = CYCLE.

## R

### Ratio control

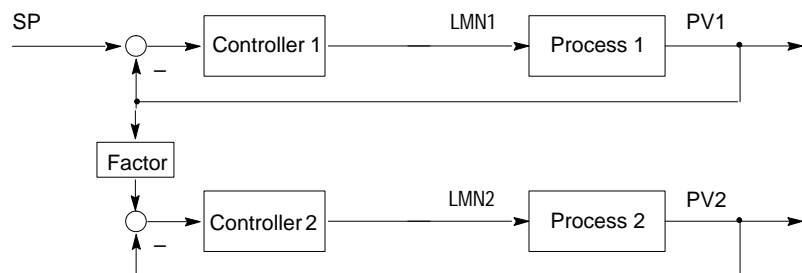
- Single loop ratio controller

A single loop ratio controller is used when the ratio of two process variables is more important than the absolute values of the variables.



- Multi-loop ratio controller

In a multi-loop ratio controller, the ratio of the two process variables PV1 and PV2 must be kept constant. To do this, the setpoint of the 2nd control loop is calculated from the process variable of the 1st control loop. Even if the process variable PV1 changes dynamically, the ratio is maintained.



### Remote Device

Remote devices are devices, for example printers or computers that are obtainable on a network. In contrast to local devices, they must be assigned a network address when they are installed.

### Restart

When a CPU starts up (for example, when the mode selector is moved from STOP to RUN or when the power is turned on), before cyclic program processing starts (OB1), either the organization block OB100 (complete restart) or the organization block OB101 (restart; only in the S7-400) is processed first. In a restart the process-image input table is read in and the STEP 7 user program processing is restarted at the point where it was interrupted by the last stop (STOP, power off).

<b>Result of logic operation (RLO)</b>	The result of logic operation (RLO) is the current signal state in the processor which is used for further binary signal processing. The signal state of the last RLO decides whether or not certain operations are executed.
<b>Run-time error</b>	Errors which occur during execution of the user program in the PLC (not in the process).
<b>S</b>	
<b>Standard function</b>	Standard functions are → function blocks available from SIEMENS for implementing complex tasks.
<b>Standard function block</b>	Standard function blocks are → function blocks available from SIEMENS for implementing complex tasks.
<b>Start event</b>	Start events are defined events such as errors or interrupts which prompt the operating system to call the appropriate organization block.
<b>Start event information</b>	The start event information is part of an → organization block (OB). Start event information provides the S7 user with detailed information about the event which triggered the call for the OB. The start event information contains the event number (consisting of event classes and event IDs), an event time stamp, and additional information (for example, the address of the interrupt-activating signal module).
<b>Start information</b>	When the operating system calls an organization block, the operating system transfers start information which can be interpreted in the user program.
<b>Start-up OB</b>	Depending on the setting of the → start-up mode selector (only S7-400), the reason for the start-up (return of power after outage, manual switch from STOP to RUN with the → mode selector or command from the programming device) either the start-up organization block “Complete restart” or “Restart” (only exists on the S7-400) is called by the operating system. In the start-up OB, the SIMATIC S7 user can, for example, program how the system will start up again after a power outage.
<b>Statement</b>	An instruction (STEP 5 or STEP 7) is the smallest part of a program created in a textual language. It represents a command for the processor.

<b>Statement List</b>	The Statement List is the assembly language of → STEP 7. When a program is processed in STL, the individual instructions correspond to the sequence with which the CPU processes the program.
<b>STEP 7</b>	Programming software for creating → user programs for SIMATIC S7 controllers.
<b>STEP 7 programming language</b>	Programming language for SIMATIC S7 controllers. The S7 programmer can use STEP 7 in different representation types: a) Statement List, b) Control System Flowchart, c) Ladder Logic.
<b>Step controller</b>	A step controller is a quasi continuous controller with a discontinuous output (and motor-driven actuator with an I action). The actuator has a three-step response, for example up – stop – down (or open – hold – close) (→ Three-step controller).
<b>STL</b>	→ Statement List.
<b>Symbolic programming</b>	<p>The STEP 7 programming language allows the use of symbolic names instead of STEP 7 addresses. For example, a STEP 7 address “Q 1.1” can be replaced with “Valve 17”.</p> <p>The symbol list in STEP 7 also creates the link between the → address and the assigned symbolic name.</p>
<b>System function (SFC)</b>	A system function (SFC) is a function which is integrated in the CPU operating system and can be called in the STEP 7 user program as required.
<b>System function block (SFB)</b>	A system function block (SFB) is a → function block integrated in the CPU operating system which can be called in the STEP 7 user program when required.
<b>T</b>	
<b>Three-step controller</b>	A controller that can only adopt three discrete states; for example “heat – off cool” or “right – stop – left” (→ Step controller).
<b>Two-step controller</b>	A two-step controller is a controller that can only set two states for the manipulated variable (for example, on – off).

**U**

**User program** The user program contains all the → statements and → declarations and the data for signal processing with which a system or process can be controlled. It is assigned to a programmable module (→ Module, programmable) and can be structured in smaller units known as → blocks.

**User program error** Errors which may occur during the processing of the user program in a SIMATIC S7 PLC (in contrast to → process errors). The operating system handles errors using error OBs ( → priority class system), the → status word and output parameters from → system functions.

**V**

**Variable** A variable defines a data with a variable content that can be used in the STEP 7 user program. A variable consists of an address (for example, M 3.1) and a data type (for example, BOOL) and is represented by a symbol (for example, MOTOR\_ON).

**Variable declaration** The variable declaration incorporates the entry of a symbolic name, a data type and possibly a default value, address and comment.





# Alphabetical Index

## A

- Access error, 10-2, 10-9
- Access error filter, 10-5
- ACCFLT\_ESR, 10-12
- ACCFLT\_MASKED, 10-10, 10-11
- ACCFLT\_QUERY, 10-12
- ACCFLT\_RESET\_MASK, 10-11
- ACCFLT\_SET\_MASK, 10-10
- ACT\_TINT, 8-7
- ADC/DAC error, A-3
- ALARM, 19-6
- ALARM\_8, 19-11
- ALARM\_8P, 19-9
- ALARM\_S, 19-23
- ALARM\_SC, 19-26
- ALARM\_SQ, 19-23
- Alignment error
  - when reading, 10-7
  - when writing, 10-7
- AR\_SEND, 19-13
- Area error
  - when reading, 10-7
  - when writing, 10-7
- Area length error
  - when reading, 10-7
  - when writing, 10-7
- Asynchronous error, 11-2, C-6
  - delaying with SFC41 DIS\_AIRT, 11-8
  - disabling with SFC39 DIS\_IRT, 11-4
  - enabling with SFC40 EN\_IRT, 11-6
  - enabling with SFC42 EN\_AIRT, 11-9
- Asynchronous error interrupts, 1-4
- Asynchronous errors, OB80, 1-19–1-22

## B

- Battery backup, failed, A-3
- Battery exhausted, A-3
- BCD conversion error, 10-7
- Binary selection, 21-26
- Bit field in the I/O area
  - resetting with SFC80, 13-5
  - setting with SFC79, 13-4

- BLK, 3-5
- BLKMOV, 3-2
- Block number error, 10-8
- Block types, B-13
- BRCV, 17-12
- BSEND, 17-10
- BVAL, 3-5

## C

- CAN\_DINT, 9-6
- CAN\_TINT, 8-6
- CDT, 5-3
- Channel
  - error, A-3, A-4
  - information, A-3
  - type, A-4
- Clock
  - master, 5-2
  - synchronization, 5-2
- Clocks, synchronization of, 5-2
- Common mode error
  - analog input module, A-5
  - analog output module, A-6
- Communication
  - error, 11-3
  - interrupt, 11-2
  - performance parameters, B-28
  - problem, A-3
  - status data, B-42
- Communication events, C-13
- Communication SFBs. *See* SFBs
- Communication SFCs, 18-4
- Communication SFCs for non-configured connections, classification, 18-4
- Complete restart, 1-39
  - initiating on a remote device, 17-25
- COMPRESS, 3-11
- Configuration, error
  - analog input module, A-5
  - analog output module, A-6
  - digital input module, A-6
  - digital output module, A-7

CONT\_C, 22-4  
 CONT\_S, 22-11  
 CONTROL, 17-35  
 Control  
     continuous control with SFB41, 22-4  
     step control with SFB42, 22-11  
 Copying variables, with SFC20 BLKMOV, 3-2  
 COUNT, 3-6  
 Count down, 20-6  
 Count up, 20-5  
 Count up and down, 20-7  
 Counters, number error, 10-7  
 CPU  
     changing to the STOP mode with SFC46  
         STP, 4-3  
     characteristics, B-8  
     hardware fault, 11-3  
 CPU hardware fault OB, 1-27  
 CQ, 6-5  
 CREATE\_DB, 3-6  
 CTD, 20-6  
 CTRL\_RTM, 6-4  
 CTU, 20-5  
 CTUD, 20-7  
 Current below measuring range, analog input  
     module, A-5  
 Current supply monitoring, analog output mod-  
     ule, A-6  
 CV, 6-5  
 Cycle time monitoring, A-3  
 Cyclic interrupts, 11-2  
     OB35, 1-13

## D

Data block  
     creating with SFC22 CREAT\_DB, 3-6  
     deleting with SFC23, 3-8  
     testing with SFC24, 3-10  
 Data record  
     reading, 7-2  
     reading with SFC59 RD\_REC, 7-9  
     writing, 7-2  
     writing with SFC58 WR\_REC, 7-8

Data type conversion  
     DINT to STRING, 21-21  
     INT to STRING, 21-21  
     REAL to STRING, 21-22  
     STRING to DINT, 21-23  
     STRING to INT, 21-22  
     STRING to REAL, 21-23  
 DATE, extract from DATE\_AND\_TIME, 21-5  
 Date, 5-2  
 DATE and TIME\_OF\_DAY, combine to  
     DATE\_AND\_TIME, 21-5  
 DATE\_AND\_TIME  
     compare for equal, 21-10  
     compare for greater than, 21-11  
     compare for greater than or equal, 21-10  
     compare for less than, 21-12  
     compare for less than or equal, 21-11  
     compare for unequal, 21-12  
 Day of the week, extract from  
     DATE\_AND\_TIME, 21-6  
 DB\_NUMBER, 3-6  
 DEL\_DB, 3-8  
 Delay time, 9-3  
 Delaying the user program, with SFC47 WAIT,  
     4-4  
 Diagnostic buffer, 10-2, B-2, B-64  
 Diagnostic data, A-2  
     content, A-2  
     of the CPU, B-2  
     of the signal modules, 7-2, B-2  
     structure, A-2  
 Diagnostic data of a module, A-2  
 Diagnostic events, C-15  
 Diagnostic interrupt, 11-3  
     from substitute, A-3  
 Diagnostics of logged-on devices, B-55  
 DIS\_AIRT, 11-8  
 DIS\_IRT, 11-4  
 DIS\_MSG, 19-15  
 DMSK\_FLT, 10-11  
 DP\_PRAL, 15-2  
 DPNRM\_DG, 15-4  
 DPRD\_DAT, 15-7  
 DPWR\_DAT, 15-9

DRUM, 13-6  
 DSTBLK, 3-3  
 DTIME, 9-4  
 Duration  
   add to a time, 21-8  
   subtract from a time, 21-8

## E

EN\_AIRT, 11-9  
 EN\_IRT, 11-6  
 EN\_MSG, 19-17  
 Error  
   access, 10-2  
   ADC/DAC, A-3  
   asynchronous, 11-2  
   EPROM, A-3  
   masking, 10-2  
   programming, 10-2  
   RAM, A-3  
   synchronous, 10-2  
 Error detection, types of OB  
   OB1, 1-5  
   OB10, 1-7–1-11  
   OB121, 1-42  
   OB122, 1-45  
   OB20, 1-11  
   OB35, 1-13  
   OB80, 1-19–1-22  
 Error filter  
   access errors, 10-5  
   programming errors, 10-4  
 Error handling, 10-2  
 Error information, 2-2  
   general, 2-4  
   SFC22 CREAT\_DB, 3-7  
   SFC23 DEL\_DB, 3-9  
   SFC34 QRY\_DINT, 9-5  
   SFC36 MSK\_FLT, 10-10  
   SFC37 DMSK\_FLT, 10-11  
   SFC38 READ\_ERR, 10-12  
   SFC39 DIS\_IRT, 11-5  
   SFC40 EN\_IRT, 11-7  
   SFC42 EN\_AIRT, 11-9  
   specific, 2-4  
 Error interrupt  
   asynchronous, 11-2  
   synchronous, 11-2

Error OB, 10-2  
   types of OB  
     OB1, 1-5  
     OB20, 1-11  
     OB35, 1-13  
     OB80, 1-19–1-22  
     OB81, 1-21  
     OB82, 1-23  
     OB85, 1-28–1-32  
 Error register, 10-2  
   reading with SFC38 READ\_ERR, 10-12  
 Event, C-2  
   class, C-2  
   ID, 12-12, C-2  
 EVENTN, 12-13  
 Existing priority classes, B-14  
 Expansion rack failure, A-3  
 External error, A-3

## F

FC1, 21-8  
 FC10, 21-13  
 FC11, 21-20  
 FC12, 21-10  
 FC13, 21-13  
 FC14, 21-11  
 FC15, 21-14  
 FC16, 21-21  
 FC17, 21-18  
 FC18, 21-11  
 FC19, 21-14  
 FC2, 21-18  
 FC20, 21-16  
 FC21, 21-16  
 FC22, 21-24  
 FC23, 21-12  
 FC24, 21-15  
 FC25, 21-24  
 FC26, 21-17  
 FC27, 21-25  
 FC28, 21-12  
 FC29, 21-15  
 FC3, 21-5  
 FC30, 21-22  
 FC31, 21-19  
 FC32, 21-17  
 FC33, 21-7

FC34, 21-9  
FC35, 21-8  
FC36, 21-26  
FC37, 21-23  
FC38, 21-22  
FC39, 21-23  
FC4, 21-19  
FC40, 21-7  
FC5, 21-21  
FC6, 21-5  
FC7, 21-6  
FC8, 21-6  
FC9, 21-10  
FILL, 3-4  
Filter, errors, 10-3  
Free user events, C-19  
Fuse tripped, A-3

## G

GADR\_LGC, 14-2  
GD packet  
    fetching with SFC61, 16-4  
    sending with SFC60, 16-2  
GD\_RCV, 16-4  
GD\_SND, 16-2  
GET, 17-14  
Ground error  
    digital input module, A-6  
    digital output module, A-7

## H

Hardware interrupt, 11-2  
    lost, A-3  
Hardware interrupt OBs, 1-15

## I

I/O access error  
    when reading, 10-9  
    when writing, 10-9  
I\_ABORT, 18-26  
I\_GET, 18-22  
I\_PUT, 18-24  
IEC functions  
    overview of, 21-2  
    work and load memory requirements, 21-3  
INFO1, 12-13  
INFO2, 12-13

Initializing a memory area, with SFC21 FILL, 3-4

Integrated control  
    analysis of the process, 22-2  
    applications, 22-2  
    selecting a controller, 22-2

Internal error, A-3

Interrupt, 11-2  
    and errors, assignment to OBs, B-18  
    classes, 11-2  
    cyclic (OB35), 1-13  
    delaying with SFC41 DIS\_AIRT, 11-8  
    disabling with SFC39 DIS\_IRT, 11-4  
    enabling with SFC40 EN\_IRT, 11-6  
    enabling with SFC42 EN\_AIRT, 11-9  
    time-delay (OB20), 1-11

Interrupt OBs  
    diagnostic interrupts, 1-23  
    insert / remove module interrupt, 1-25  
    time-of-day interrupts, 1-7

Interrupt status, B-20

Interrupts, hardware interrupt OBs, 1-15

IOID, 7-4, 7-5, 7-6, 7-8, 7-9

## L

LADDR, 7-4, 7-5, 7-6, 7-8, 7-9

LGC\_GADR, 14-4

Limit, 21-24

Local data of the OBs, B-56

Logical address  
    of a channel, querying, 14-2  
    of a module, querying all addresses, 14-6

LOW\_LIMIT, 3-6

## M

M short circuit  
    analog input module, A-5  
    analog output module, A-6  
    digital input module, A-6  
    digital output module, A-7  
Main program cycle (OB1), 1-5  
Masking, errors, 10-2  
Master clock, 5-2  
Measuring range exceeded, analog input module, A-5  
Memory areas, B-10  
Memory card, A-3  
MODE, 11-4, 11-6

Mode, A-3  
 Mode run-time events, C-11  
 Module  
   class, A-3  
   fault, A-3  
   identification, B-7  
   type ID, B-4  
 Module diagnostic data, B-66, B-67  
 Module diagnostic information, B-65  
 Module slot, of a logical address, querying, 14-4  
 Module status information, B-58  
 MP\_ALM, 4-5  
 MSK\_FLT, 10-10  
 Multicomputing interrupt, 11-2

## N

No auxiliary voltage, A-3  
 No front connector, A-3  
 No load voltage  
   analog output module, A-6  
   digital output module, A-7  
 No parameter assignment, A-3  
 No sensor power supply, digital input module, A-6  
 NOTIFY, 19-4  
 NR, 6-3, 6-4, 6-5

## O

OB\_NR, 8-5, 8-6, 8-7, 8-8, 9-4, 9-5, 9-6, 11-4, 11-6  
 Off delay, generating, 20-4  
 On delay, generating, 20-3  
 Operating mode, B-24  
 Operating mode transition, B-24

Organization block (OB), 1-2  
   background OB (OB90), 1-37  
   multicomputing interrupt OB (OB60), 1-17  
   OB1, 1-5  
   OB121, 1-42  
   OB122, 1-45  
   types of  
     OB20, 1-11  
     OB35, 1-13  
     OB80, 1-19–1-22  
     OB81, 1-21  
     OB82, 1-23  
     OB85, 1-28–1-32

Organization blocks (OBs)  
   communication error OB (OB87), 1-35  
   complete restart OB (OB100), 1-39  
   CPU hardware fault OB (OB84), 1-27  
   diagnostic interrupt OB (OB82), 1-23  
   insert / remove module interrupt OB (OB83), 1-25  
   overview, 1-2, 1-3  
   priority class error OB (OB85), 1-28  
   rack failure OB (OB86), 1-31  
   restart OB (OB101), 1-39  
   start-up OBs (OBs 100 and 101), 1-39  
   time-of-day interrupt OBs (OBs 10 to 17), 1-7

Overtemperature, digital output module, A-7

## P

P short circuit  
   analog input module, A-5  
   analog output module, A-6  
   digital input module, A-6  
   digital output module, A-7

## Parameter

ACCFLT\_ESR, 10-12  
 ACCFLT\_MASKED, 10-10  
 ACCFLT\_Masked, 10-11  
 ACCFLT\_QUERY, 10-12  
 ACCFLT\_RESET\_MASK, 10-11  
 ACCFLT\_SET\_MASK, 10-10  
 BLK, 3-5  
 BUSY with SFCs 51 and 55 to 59, 2-8  
 BVAL, 3-5  
 CDT, 5-3  
 COUNT, 3-6  
 CQ, 6-5  
 CV, 6-5  
 DB\_NUMBER, 3-6  
 DSTBLK, 3-3  
 DTIME, 9-4  
 EVENTN, 12-13  
 INFO1, 12-13  
 INFO2, 12-13  
 IOID, 7-4, 7-5, 7-6, 7-8, 7-9  
 LADDR, 7-4, 7-5, 7-6, 7-8, 7-9  
 LOW\_LIMIT, 3-6  
 MODE, 11-4, 11-6  
 NR, 6-3, 6-4, 6-5  
 OB\_NR, 8-5, 8-6, 8-7, 8-8, 9-4, 9-5, 9-6,  
     11-4, 11-6  
 PDT, 5-2  
 PERIOD, 8-5  
 PRGFLT\_ESR, 10-12  
 PRGFLT\_MASKED, 10-10, 10-11  
 PRGFLT\_QUERY, 10-12  
 PRGFLT\_RESET\_MASK, 10-11  
 PRGFLT\_SET\_MASK, 10-10  
 PV, 6-3  
 RECNUM, 7-4, 7-5, 7-8, 7-9  
 RECORD, 7-4, 7-8  
 REQ with asynchronous SFCs, 2-7  
 RET\_VAL with SFCs 51 and 55 to 59, 2-8  
 SDT, 8-5  
 SEND, 12-13  
 SRCBLK, 3-3  
 STATUS, 8-8, 9-5  
 UP\_LIMIT, 3-6  
 write default parameter, 7-5  
 wrong parameter in module, A-3  
 WT, 4-4

## Parameter assignment error

analog input module, A-5  
 analog output module, A-6  
 digital input module, A-6  
 digital output module, A-7

## Parameters

of the signal modules, 7-2  
 SFC1 READ\_CLK, 5-3  
 SFC21 FILL, 3-5  
 SFC22 CREATE\_DB, 3-6  
 SFC29 CAN\_TINT, 8-6  
 SFC34 QRY\_DINT, 9-5  
 SFC37 DMSK\_FLT, 10-11  
 SFC38 READ\_ERR, 10-12  
 SFC39 DIS\_IRT, 11-4  
 SFC4 READ\_RTM, 6-5  
 SFC40 EN\_IRT, 11-6  
 SFC41 DIS\_AIRT, 11-8  
 SFC42 EN\_AIRT, 11-9  
 SFC47 WAIT, 4-4  
 SFC52 WR\_USRMSG, 12-13  
 SFC57 PARM\_MOD, 7-6  
 SFC58 WR\_REC, 7-8  
 SFC64 TIME\_TICK, 6-6  
 PARM\_MOD, 7-6  
 PDT, 5-2  
 PERIOD, 8-5  
 Power supply, failed, A-3  
 Power supply error, 11-3  
 PRGFLT\_ESR, 10-12  
 PRGFLT\_MASKED, 10-10, 10-11  
 PRGFLT\_QUERY, 10-12  
 PRGFLT\_RESET\_MASK, 10-11  
 PRGFLT\_SET\_MASK, 10-10  
 PRINT, 17-18  
 Priority class, 1-2, 1-3, 1-6, 1-10, 1-12, 1-14,  
     1-15, 1-16, 1-18, 1-19, 1-21, 1-23, 1-25,  
     1-27, 1-28, 1-31, 1-35, 1-38, 1-39, 1-43,  
     1-45, 10-12, 12-3, 12-7, 12-8, B-5, B-14,  
     B-19, B-21, B-23, C-8, C-10  
 types of OB  
     OB1, 1-5  
     OB121, 1-42  
     OB122, 1-45  
     OB20, 1-11  
     OB35, 1-13  
     OB80, 1-19–1-22  
     OB81, 1-21  
     OB82, 1-23  
     OB85, 1-28–1-32  
 Priority class error OB, 1-28  
 Priority classes  
     existing, B-14  
     status of, B-22  
 Processor failure, A-3  
 Program, cyclic, 1-5  
 Program error, 11-3

## Programming, types of OB

- OB1, 1-5
- OB121, 1-42
- OB122, 1-45
- OB20, 1-11
- OB35, 1-13
- OB80, 1-19–1-22
- OB81, 1-21
- OB82, 1-23
- OB85, 1-28–1-32

Programming error, 10-2, 10-7

Programming error filter, 10-4

Pulse, generating, 20-2

Pulse duration modulation, 22-17

PULSEGEN, 22-17

PUT, 17-16

PV, 6-3

## Q

QRY\_DINT, 9-5

QRY\_TINT, 8-8

## R

Rack failure, 1-31, 11-3

RAM error, A-3

RD\_LGADR, 14-6

RD\_REC, 7-9

RD\_SINFO, 12-2

RDSYSST, 12-4, B-2

RE\_TRIGR, 4-2

READ\_CLK, 5-3

READ\_ERR, 10-12

READ\_RTM, 6-5

Reading

- consistent data of a DP slave, 15-7

- diagnostic data of a DP slave, 15-4

- with SFC51 RDSYSST, 12-4

Reading data from a remote CPU with SFB14, 17-14

Reading OB start information with SFC6, 12-2

Reading the system status, with SFC51

- RDSYSST, 12-4

Reading the system time, with SFC64

- TIME\_TCK, 6-6

Reading the time, with SFC1 READ\_CLK, 5-3

Receiving segmented data, with SFB 13, 17-12

RECNUM, 7-4, 7-5, 7-8, 7-9

RECORD, 7-4, 7-8

Reference channel error, analog input module, A-5

Remove/insert module interrupt, 11-3

REPL\_VAL, 3-13

Restart, 1-39

- initiating on a remote device, 17-29

RESUME, 17-29

Retriggering cycle time monitoring, with SFC43

- RE\_TRIGR, 4-2

Return value

- SFC41 DIS\_AIRT, 11-8

- SFC42 EN\_AIRT, 11-9

Run-time meter, 6-2

- characteristics, 6-2

- range of values, 6-2

- reading out with SFC4 READ\_RTM, 6-5

- setting with SFC2 SET\_RTM, 6-3

- starting with SFC3 CTRL\_RTM, 6-4

- stopping with SFC3 CTRL\_RTM, 6-4

## S

S, 6-4

S5TIME, data type format to TIME, 21-7

S7-300 I/O configuration, B-16

SBDs, permitted, B-15

SDT, 8-5

Select maximum, 21-25

Select minimum, 21-25

SEND, 12-13

Sending segmented data, with SFB 12, 17-10

Sequencer, implementing, 13-6

SET, 13-4, 13-5

SET\_CLK, 5-2

SET\_RTM, 6-3

SET\_TINT, 8-5

Setting the time, with SFC0 SET\_CLK, 5-2

SFB0 CTU, 20-5

SFB1 CTD, 20-6

SFB12 BSEND, 17-10

SFB13 BRCV, 17-12

SFB14 GET, 17-14

SFB15 PUT, 17-16

SFB16 PRINT, 17-18

SFB19 START, 17-25

SFB2 CTUD, 20-7

SFB20 STOP, 17-27

SFB21 RESUME, 17-29

SFB22 STATUS, 17-31

SFB23 USTATUS, 17-33

- SFB3 TP, 20-2
- SFB32 DRUM, 13-6
- SFB33 ALARM, 19-6
- SFB34 ALARM\_8, 19-11
- SFB35 ALARM\_8P, 19-9
- SFB36 NOTIFY, 19-4
- SFB37 AR\_SEND, 19-13
- SFB4 TON, 20-3
- SFB41 CONT\_C, 22-4
  - block diagram, 22-6
- SFB42 CONT\_S, 22-11
  - block diagram, 22-13
- SFB43 PULSEGEN, 22-17
  - automatic synchronization, 22-18
  - block diagram, 22-18
  - three-step control, 22-21
  - three-step control asymmetrical, 22-22
  - two-step control, 22-23
- SFB5 TOF, 20-4
- SFB8 USEND, 17-7
- SFB9 URCV, 17-8
- SFBs
  - classification of, 17-2
  - parameter classification, 17-3
  - querying the status of a connection belonging to an SFB instance, 17-35
  - reaction to startup, 17-37
  - reactions to errors and faults, 17-39
- SFC 59 RD\_REC, 7-13
- SFC 65 X\_SEND, 18-11
- SFC 7 DP\_PRAL, 15-2
- SFC0 SET\_CLK, 5-2
- SFC1 READ\_CLK, 5-3
  - parameters, 5-3
- SFC10 DIS\_MSG, 19-15
- SFC13 DPNRM\_DG, 15-4
- SFC14 DPRD\_DAT, 15-7
- SFC15 DPWR\_DAT, 15-9
- SFC17 ALARM\_SQ, 19-23
- SFC18 ALARM\_S, 19-23
- SFC19 ALARM\_SC, 19-26
- SFC2 SET\_RTM, 6-3
- SFC20 BLKMOV, 3-2
- SFC21 FILL, 3-4
  - parameters, 3-5
- SFC22 CREAT\_DB, error information, 3-7
- SFC22 CREATE\_DB, 3-6
  - parameters, 3-6
- SFC23 DEL\_DB, 3-8
  - error information, 3-9
- SFC24 TEST\_DB, 3-10
- SFC25 COMPRESS, 3-11
- SFC26 UPDAT\_PI, 13-2
- SFC27 UPDAT\_PO, 13-3
- SFC28 SET\_TINT, 8-5
- SFC29 CAN\_TINT, 8-6
  - parameters, 8-6
- SFC3 CTRL\_RTM, 6-4
- SFC30 ACT\_TINT, 8-7
- SFC31 QRY\_TINT, 8-8
- SFC32 SRT\_DINT, 9-4
- SFC33 CAN\_DINT, 9-6
- SFC34 QRY\_DINT, 9-5
  - error information, 9-5
  - parameters, 9-5
- SFC35 MP\_ALM, 4-5
- SFC36 MSK\_FLT, 10-10
  - error information, 10-10
- SFC37 DMSK\_FLT, 10-11
  - error information, 10-11
  - parameters, 10-11
- SFC38 READ\_ERR, 10-12
  - error information, 10-12
  - parameters, 10-12
- SFC39 DIS\_IRT, 11-4
  - error information, 11-5
  - parameters, 11-4
- SFC4 READ\_RTM, 6-5
  - parameters, 6-5
- SFC40 EN\_IRT, 11-6
  - error information, 11-7
  - parameters, 11-6
- SFC41 DIS\_AIRT, 11-8
  - parameters, 11-8
  - return value, 11-8
- SFC42 EN\_AIRT, 11-9
  - error information, 11-9
  - parameters, 11-9
  - return value, 11-9
- SFC43 RE\_TRIGR, 4-2
- SFC44 REPL\_VAL, 3-13
- SFC46 STP, 4-3
- SFC46 TIME\_TCK, 6-6
- SFC47 WAIT, 4-4
  - parameters, 4-4
- SFC48 SNC\_RTCB, 5-4
- SFC49 LGC\_GADR, 14-4
- SFC5 GADR\_LGC, 14-2
- SFC50 RD\_LGADR, 14-6
- SFC51 RDSYSST, 12-4, B-2
- SFC52 WR\_USRMSG, 12-11
  - parameters, 12-13
- SFC55 WR\_PARM, 7-4
- SFC56 WR\_DPARM, 7-5



SFC57 PARM\_MOD, 7-6  
 parameters, 7-6  
 SFC58 WR\_REC, 7-8  
 parameters, 7-8  
 SFC59 RD\_REC, 7-9  
 SFC6 RD\_SINFO, 12-2  
 SFC60 GD\_SND, 16-2  
 SFC61 GD\_RCV, 16-4  
 SFC62 CONTROL, 17-35  
 SFC64 TIME\_TICK, parameters, 6-6  
 SFC66 X\_RCV, 18-13  
 SFC67 X\_GET, 18-17  
 SFC68 X\_PUT, 18-19  
 SFC69 X\_ABORT, 18-21  
 SFC72 I\_GET, 18-22  
 SFC73 I\_PUT, 18-24  
 SFC74 I\_ABORT, 18-26  
 SFC79 SET, 13-4, 13-5  
 SFC9 EN\_MSG, 19-17  
 SIGN, 9-4  
 Slave clocks, synchronization of, 5-4  
 SNC\_RTCB, 5-4  
 SRCBLK, 3-3  
 SRT\_DINT, 9-4  
 Standard OB events, C-3  
 Standard user events, C-17  
 START, 17-25  
 Start events, B-57  
 Startup, 1-39  
 STATUS, 8-8, 9-5, 17-31  
 Status of a remote partner  
 querying, 17-31  
 receiving, 17-33  
 STEP 7, types of OB  
 OB1, 1-5  
 OB10, 1-7–1-11  
 OB121, 1-42  
 OB122, 1-45  
 OB20, 1-11  
 OB35, 1-13  
 OB80, 1-19–1-22  
 OB81, 1-21  
 OB82, 1-23  
 OB85, 1-28–1-32  
 STOP, 17-27  
 changing a remote device to, 17-27  
 Stop and abort events, C-8  
 STP, 4-3

STRING  
 combine two STRING variables, 21-18  
 compare for equal, 21-13  
 compare for greater than, 21-14  
 compare for greater than or equal, 21-13  
 compare for less than, 21-15  
 compare for less than or equal, 21-14  
 compare for unequal, 21-15  
 delete in a STRING variable, 21-19  
 find in a STRING variable, 21-20  
 insert in a STRING variable, 21-18  
 left part of, 21-16  
 length, 21-16  
 middle part of, 21-17  
 replace in a STRING variable, 21-19  
 right part of, 21-17  
 Substitute value, writing to ACCU 1 with  
 SFC44 REPL\_VAL, 3-13  
 Supply current monitoring, analog input module, A-5  
 Synchronous error interrupts, 1-4  
 Synchronous errors, 10-2, C-5  
 masking with SFC36 MSK\_FLT, 10-10  
 OB121, 1-42  
 OB122, 1-45  
 unmasking with SFC37 DMSK\_FLT, 10-11  
 System areas, B-12  
 System data, B-2  
 System data blocks, permitted, B-15  
 System status list, B-2  
 partial lists, B-5  
 SZL-ID, B-4  
 SZL\_HEADER, 12-5

## T

Temporary variables (TEMP), required for OBs,  
 1-6, 1-45  
 TEST\_DB, 3-10  
 TIME, data type format to S5TIME, 21-7  
 Time, subtract two time values, 21-9  
 Time error, 11-3  
 Time of day (TOD), 5-2

Time-delay interrupt, 9-2, 11-2  
     canceling with SFC33 CAN\_DINT, 9-6  
     conditions for the call, 9-2  
     querying with SFC34 QRY\_DINT, 9-5  
     situations affecting, 9-2  
     starting in the start-up OB, 9-3  
     starting with SFC32 SRT\_DINT, 9-4  
 Time-of-day interrupt, 11-2  
 TIME\_OF\_DAY, extract from  
     DATE\_AND\_TIME, 21-6  
 TIME\_TCK, 6-6  
 Time-delay interrupts, OB20, 1-11  
 Time-of-day interrupt, 8-2  
     activating with SFC30 ACT\_TINT, 8-7  
     canceling with SFC29 CAN\_TINT, 8-6  
     complete restart, 8-4  
     conditions for the call, 8-2  
     execution and reaction, 8-4  
     OB, 8-2  
     querying with SFC31 QRY\_TINT, 8-8  
     setting with SFC28 SET\_TINT, 8-5  
     situations affecting, 8-3  
 Timer number error, 10-7  
 TOF, 20-4  
 TON, 20-3  
 TP, 20-2  
 Transferring parameters  
     with SFC55 WR\_PARM, 7-4  
     with SFC56 WR\_DPARM, 7-5  
 Type ID, of a module, B-4

## U

Uncoordinated receiving of data, with SFB 9,  
     17-8  
 Uncoordinated sending of data, with SFB8, 17-7  
 Unmasking, error events, 10-2  
 UP\_LIMIT, 3-6  
 UPDAT\_PI, 13-2  
 UPDAT\_PO, 13-3  
 Updating the process image input table, 13-2  
 Updating the process image output table, 13-3  
 URCV, 17-8  
 USEND, 17-7

User information, A-3  
 User memory, compressing with SFC25, 3-11  
 USTATUS, 17-33

## V

Variable declaration table  
     for OB1, 1-5  
     for OB10, 1-7–1-11  
     for OB121, 1-42  
     for OB122, 1-45  
     for OB20, 1-11  
     for OB35, 1-13  
     for OB80, 1-19–1-22

## W

WAIT, 4-4  
 Wire break  
     analog input module, A-5  
     analog output module, A-6  
     digital input module, A-6  
     digital output module, A-7  
 WR\_DPARM, 7-5  
 WR\_PARM, 7-4  
 WR\_REC, 7-8  
 WR\_USRMSG, 12-11  
 Write, consistent data to a DP standard slave,  
     15-9  
 Write error  
     data block, 10-8  
     instance block, 10-8  
 Writing data to a remote CPU with SFB15,  
     17-16  
 WT, 4-4

## X

X\_ABORT, 18-21  
 X\_GET, 18-17  
 X\_PUT, 18-19  
 X\_RCV, 18-13  
 X\_SEND, 18-11

Siemens AG  
AUT E 146

Östliche Rheinbrückenstr. 50  
D-76181 Karlsruhe  
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:

- |  |  |
|--|--|
| <input type="checkbox"/> Automotive              | <input type="checkbox"/> Pharmaceutical  |
| <input type="checkbox"/> Chemical                | <input type="checkbox"/> Plastic         |
| <input type="checkbox"/> Electrical Machinery    | <input type="checkbox"/> Pulp and Paper  |
| <input type="checkbox"/> Food                    | <input type="checkbox"/> Textiles        |
| <input type="checkbox"/> Instrument and Control  | <input type="checkbox"/> Transportation  |
| <input type="checkbox"/> Nonelectrical Machinery | <input type="checkbox"/> Other _ _ _ _ _ |
| <input type="checkbox"/> Petrochemical           |  |



## 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.

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?                    | <input type="checkbox"/> |
| 2. | Is the information you need easy to find?                  | <input type="checkbox"/> |
| 3. | Is the text easy to understand?                            | <input type="checkbox"/> |
| 4. | Does the level of technical detail meet your requirements? | <input type="checkbox"/> |
| 5. | Please rate the quality of the graphics/tables:            | <input type="checkbox"/> |

Additional comments:

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----

-----