SIEMENS

S5-155H
Programmable Controller
(CPU 948R)

SIMATIC S5

S5-155H Programmable Controller (CPU 948R)

Manual Volume 1/2

Order No. 6ES5 998-4SR21
Release 02

Preface (How to Use This Manual)
Introduction to the Installation and Operation of the S5-155H 1
H-Specific System Functions 2
CPU 948R 3
I/O Operating Modes and Permissible I/O Modules 4
Operating CPs/IPs in the S5-155H 5
Installation and Startup 6
Time Characteristics of the S5-155H 7
S5-155H Error Diagnostics 8
Dynamic Response to Faults, Repairs, Replacement and Upgrading 9
Typical Applications 10
Technical Specifications: IM 314R / IM 324R 11
Glossary 12

Data Handling Blocks
for the CPU 948R

Preface (How to Use This Manual)
Using the Data Handling Blocks 1
Assigning Parameters to the Data Handling Blocks 2
Data Handling Blocks in the User Program 3
Description: Data Handling Blocks 4

COM 155 H
Programmer Software for
Configuring the S5-155H
Programmable Controller

Preface (How to Use This Manual)
Working with COM 155H 1
Main Menu 2
Configuring and Initializing 3
Configuring the I/O 4
Error Diagnostics and Documentation 5
Abbreviations, Index

Copyright © Siemens AG 1994 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.

Technical data subject to change

Siemens AG
Automation Group
EWK Elektronikwerk Karlsruhe

6ES5 998-4SR21

SIEMENS

SIMATIC S5

S5-155H Programmable Controller (CPU 948R)

Instructions (S5-155H, Part I)

C79000-B8576-C197-02

Preface (How to Use This Manual)

Introduction to Installation and Operation of the S5-155H 1

H-Specific System Functions 2

CPU 948R 3

I/O Operating Modes and Permissible I/O Modules 4

CP/IP Operation in the S5-155H 5

Installation and Startup 6

Time Characteristics of the S5-155H 7

S5-155H Error Diagnostics 8

Dynamic Response to Faults, Repair, Replacement and Upgrading 9

Sample Applications 10

Technical Specifications: IM 314R / IM 324R 11

Glossary 12

Copyright © Siemens AG 1994 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.

Technical data subject to change

Siemens AG
Automation Group
EWK Elektronikwerk Karlsruhe

6ES5 998-4SR21

Preface (How to Use This Manual)

This manual (Volume 1) describes the hardware, startup procedures and functions of the S5-155H programmable controller with CPU 948R (R standing for redundant).

Volume 2 of this manual covers programming of the S5-155H, including the writing of the user program, and provides information regarding status, interrupt and error handling, latching functions and debugging aids.

The S5-155H differs from the standard version of the S5-155H programmable controller because of its fault tolerance, which allows it to be operated at a higher level of availability. The "H" indicates that it has a higher degree of availability than standard systems.

The S5-155H programmable controller system is distinguished by the "redundancy" of its central controller modules and, depending on how it is configured, of its I/O modules. The I/Os may even have triple redundancy. If a redundant system component fails during a process, the process can still be controlled.

The S5-155H is a 1-out-of-2-system. The system has two CPUs. Faults are specifically defined, thus making it impossible for a given fault to be transported from one subsystem to another. In order to attain a particularly high degree of availability, the input/output area should also be configured for redundancy.



Warning

The S5-155H programmable controller (H system) is not a failsafe system, despite its high availability, fault tolerance and reaction-free design.

It must not be used in installations in which dangerous operating conditions – and thus danger to persons or the environment – could occur as a result of a fault in the programmable controller (for example the highly improbable total failure of both central controllers).

In the case of such safety-related automation tasks, either a failsafe programmable controller (such as an S5-115F which has been type-tested by the TÜV (German Technical Inspectorate) must be used, or the S5-155H must be equipped with suitable interlock controls or protective systems capable of preventing the occurrence of dangerous operating conditions.

Target Group

This manual is intended for engineers, programmers and maintenance personnel with a basic knowledge of SIMATIC S5 systems. If you have any questions that are not answered in the manual, please contact your local Siemens representative.

Notes on the Contents

The following information on the contents of the various chapters is intended to simplify the use of this part of the manual.

Note

The following restrictions apply to the S5-155H programmable controller:

The S5-155H does not have multiprocessor capabilities.

When writing your STEP 5 user program for the S5-155H, refer to Section 2.2 for information on avoiding errors or problems on initial startup. If you are careful to observe the restrictions listed there, you will be able to use all STEP 5 user programs which can run on the S5-155U on the S5-155H as well.

Chapter 1: Introduction: System Description

Chapter 1 gives you an overview of the main functions and characteristics of the S5-155U, and outlines its principles of operation. You will find more detailed information in Chapter 2.

Chapter 2: H-Specific System Functions

Chapter 2 presents detailed information on typical characteristics and individual functions of the S5-155H, particularly on operating states and restart types. Additional topics covered in this chapter include "event-controlled" synchronization and the system self-test for localizing hardware failures. The information on some topics goes into quite a bit of detail, so limit your reading as your necessity or interest dictates.

Chapter 3: CPU 948R

Chapter 3 discusses the hardware and technical specifications for the CPU 948R central processing unit. Some specific topics are the module's installation, configuration, control elements and indicators.

Chapter 4: I/O Modes and Permissible Modules

Chapter 4 deals with the possible I/O modes in the S5-155U (redundant, switched, one-sided) and the modules which may be used in each case.

If you want to use redundant I/Os, you will find the module connections you require in Section 4.2. Section 4.2 also contains information on standard function blocks FB 40 and FB 43, which are used for analog value input, and on standard function block FB 41, which is used for analog value output.

Be sure to observe the instructions in this chapter when configuring and operating your I/O modules!

Chapter 5: Operating CPs/IPs in the S5-155H

Chapter 5 discusses the use of communications processors (CPs) and intelligent I/Os (IPs) in the S5-155U. It also contains all possible configurations for redundant CP operation.

Chapter 5 also discusses special features regarding the use of data handling blocks for the S5-155H, illustrating their usage in several sample STEP 5 programs.

Chapter 6: Installation and Startup

Chapter 6 discusses the procedures for installing the central controllers and I/Os as well as the IM 304/IM 324R parallel link and IM 304/IM 314R interface modules. It provides step-by-step instructions on configuring and starting your S5-155H using your COM 155H programmer software.

Chapter 6 also describes the jumper settings on the IM 304, IM 314R and IM 324R.

Chapter 7: Time Characteristics of the S5-155H

Chapter 7 deals with the S5-155H's time characteristics, most specifically those instruction execution and system program runtimes which exceed those of the S5-155U due to the S5-155H's fault tolerance.

Chapter 8: Error Diagnostics

Chapter 8 describes all available error diagnostics options for the S5-155H. It contains details on the structure of the error data block (F-DB), in which the system program enters all recognized errors, as well as a list of error codes and their meanings. It also discusses the meaning of the H flag word, from which you can read out information on the status of your PLC or enter instructions for its control.

Chapter 9: Dynamic Response to Faults, Repairs, Replacements and Upgrading

This chapter describes the response of the various modules to faults and failures and shows you how to proceed when making the necessary repairs in order to avoid interrupting operation.

Section 9.6 tells you how you can use certain on-line functions to replace or change your user program on the memory card during operation.

Chapter 10: Typical Applications

Chapter 10 contains sample applications for S5-155H configurations with all three types of I/Os. By implementing these examples, you will have a fault tolerant (H) system that you can use and expand to meet all your requirements.

Chapter 11: Technical Specifications IM 314R/IM 324R

Chapter 11 contains all the major technical specifications for the IM 314R and IM 324R interface modules and the exact pinouts of the backplane and front connectors. You will also find specifications in this chapter on the readback delay which you should take into account when configuring your digital input/output modules.

Chapter 12: Glossary

The glossary defines 155H-specific terms.

Index

The alphabetical index at the end of the manual will help you locate the most important terms in the manual.

Remarks Form

The (green) remarks form at the very end of the manual is provided for your comments and recommendations.

Training

Consult your local Siemens representative for information on training courses to aid you in becoming familiar with this product.

Note

This manual cannot cover all details and types of configuration for the programmable controller, nor can it cover all situations which can occur in installation, operation and maintenance.

If you require further information or have questions on your specific application which are not answered sufficiently here, please contact your local Siemens representative.

Reference Material

It is recommended that you have the following reference material that supports the S5-155H system:

- Catalog ST 54.4: S5-135U, S5-155U and S5-155H Programmable Controllers (Order No. E86010-K4654-A111-A6) *
- S5-135U/155U System Manual (Order No. 6ES5 998-0SH21) *
- PG 685 Programmer (Order No. 6ES5 885-0SC21)) *
- PG 710 Programmer (Order No. C79000–G8576–C170) *
- PG 730 Programmer (Order No. C79000–G8576–C173) *
- PG 750 Programmer (Order No. C79000–G8576–C750) *
- PG 770 Programmer (Order No. C79000–G8576–C770) *
- You will find a detailed introduction to STEP 5 programming and a description of the functions of the S5-155U programmable controller and its I/O modules in
  - *Automating with the SIMATIC S5-155U* by Hans Berger, Siemens AG, ISBN 3-8009-1561-8
- STEP 5 (Order No. C79000–G8576–C140) *

* Obtainable from your local Siemens representative

Contents

1 Introduction to the Installation and Operation of the S5-155H Programmable Controller I/1-1

1.1 Characteristics and Functions of the S5-155H Programmable Controller I/1-2

1.2 S5-155H Applications I/1-3

1.3 Redundant S5-155H Configuration I/1-4

1.4 Method of Operation of the S5-155H I/1-7
S5-155H-Specific Functions I/1-7
Operating States and Operating Principles of the S5-155H I/1-8
Programming I/1-9
Program Processing I/1-9
Configuring and Error Diagnostics with COM 155H I/1-10

1.5 S5-155H Hardware Configuration I/1-11
Minimum Configuration I/1-11
Maximum Configuration I/1-11

1.6 Software I/1-13
155H System Program I/1-13
STEP 5 User Program I/1-13

2 H-Specific System Functions I/2-1

2.1 Method of Operation and Operating States of the S5-155H I/2-2

2.2 Activating the Standby I/2-5

2.3 Event-Driven Synchronization I/2-8

2.4 Switchover from Standby to Master Controller I/2-10

2.5 Self-Test I/2-11
Self-Test Strategy I/2-11

3 CPU 948R I/3-1

3.1 Technical Specifications of the CPU 948R I/3-2

3.2 CPU 948R Installation and Startup Procedures I/3-3
Removing and Inserting the Module I/3-3
Control Elements and LEDs I/3-4
Startup I/3-7
CPU 948R Interfaces I/3-9
Specifications I/3-9

4 I/O Operating Modes and Permissible I/O Modules I/4-1

4.1 Overview I/4-2

4.2 Redundant I/Os (Overview) I/4-6
Interface Modules and Expansion Units I/4-6
Digital and Analog I/O Modules I/4-6
Overview of Redundant I/O Types I/4-7

4.3	Redundant Digital Inputs/Outputs (DIs/DQs)	I/4-10
	Redundant DIs without Error Locating Facility	I/4-10
	Redundant DIs with Error Locating Facility	I/4-10
	Testing the Error Locating Facility	I/4-11
	Redundant Three-Channel DIs	I/4-12
	Configuring Redundant Process Interrupts (DI 0)	I/4-14
	Redundant DQs without Error Locating Facility (LF)	I/4-14
	Redundant DQs with Error Locating Facility (DQ Type 10)	I/4-16
	Testing the Error Locating Facility	I/4-16
	Redundant DQ with Error Locating Facility and 3 Readback DIs (DQ Type 11)	I/4-18
	Testing the Error Locating Facility	I/4-18
4.4	Redundant Analog Input/Outputs (AIs/AQs)	I/4-20
	Direct I/O Access	I/4-20
	Redundant AIs: FB 40	I/4-20
	Three-Channel Redundant AIs: FB 43	I/4-25
	One-Sided Analog Outputs	I/4-30
	Redundant Analog Outputs	I/4-30
	Redundant 2-Channel AQs without Error Locating Facility	I/4-30
	Redundant 2-Channel AQs with Error Locating Facility	I/4-30
4.5	One-Sided I/Os	I/4-38
	Interface Modules and Expansion Units (EUs)	I/4-38
	Digital and Analog Input/Output Modules	I/4-39
4.6	Switched I/Os	I/4-40
	Interface Modules and Expansion Units (EUs)	I/4-41
	Digital and Analog I/O Modules	I/4-41
4.7	Hybrid I/O Configurations	I/4-42
5	**Operation of Communications Processors and Intelligent I/Os in the S5-155H**	**I/5-1**
5.1	Intelligent I/Os in the S5-155H	I/5-2
5.2	One-Sided CP/IP Configurations	I/5-3
	Communications Processors (CPs/IPs)	I/5-3
5.3	Switched CP/IP Configurations	I/5-4
	CPs/IPs	I/5-4
5.4	Redundant Communications Processor (CP) Configurations	I/5-5
6	**Installation and Startup**	**I/6-1**
6.1	Installing the S5-155H	I/6-2
6.2	Configuring the IM 304/IM 324R Parallel Link	I/6-5
6.3	Configuring One-Sided I/Os and Putting Them Into Operation	I/6-8
6.4	Configuring Switched I/Os and Putting Them Into Operation	I/6-10
6.5	Configuring Redundant I/Os and Putting Them Into Operation	I/6-15
6.6	S5-155H Responses to Faults/Errors	I/6-17

7 Time Characteristics of the S5-155H I/7-1

7.1 Statement Execution Times for the S5-155H I/7-2

7.2 System Program Runtimes I/7-3
Increased S5-155H Runtimes I/7-3
S5-155H Restart Time I/7-3

7.3 Dynamic Response on Standby Activation I/7-4

7.4 Interrupt Response Time I/7-5

8 S5-155H Error Diagnostics I/8-1

8.1 Troubleshooting and Error Handling in the S5-155H I/8-2

8.2 Error Data Block (E-DB) I/8-4
Structure of the Error DB I/8-5
Notes on Error Diagnostics I/8-23
Evaluating the Error Data Block I/8-24
Error Messages from the CP 523 ..... I/8-24

8.3 H Flag Doubleword I/8-27

8.4 Error Organization Block OB 37 ..... I/8-28

8.5 The H Flag Word I/8-29

9 Dynamic Response to Faults, Repair, Replacement and Upgrading I/9-1

9.1 Failure and Repair of the CPU and Parallel Links I/9-2
Replacing the Central Processing Unit I/9-2
Replacing the Interface Modules (IMs) I/9-2
Replacing the IM 30x Interface Modules in the CC (Excepting those used for the Parallel Link) I/9-3
Replacing the IM 30x – IM 31x Connecting Cable I/9-3

9.2 Failure and Repair of Expansion Units (EUs) I/9-4

9.3 Failure and Repair of I/O Modules I/9-5

9.4 Failure and Repair of CP/IP Modules I/9-6

9.5 Standby-Master Switchover I/9-8

9.6 Replacing the Memory Card During Operation I/9-10

9.7 Upgrading the CPU 948R's RAM or Version Number I/9-11

10 Typical Applications I/10-1

10.1 Task and Required Resources I/10-2

10.2 Installing the Hardware I/10-3
System Configuration I/10-3
Establishing a CC-to-CC Parallel Link I/10-3
Installing the I/O Bus I/10-5

10.3 Configuring Switched I/Os I/10-6
Calling COM 155H I/10-6
Initializing the Operating System (COM 155H) I/10-7
Setting the System Size I/10-8

	Configuring Digital Inputs	I/10-9
	Configuring Digital Outputs	I/10-9
	Transferring Configuring Data (DX 1)	I/10-10
	Starting Up the PLC	I/10-10
10.4	Configuring One-Sided I/Os	I/10-11
10.5	Configuring Redundant I/Os	I/10-12
10.6	Redundant Point-To-Point Link	I/10-17
	Two-Channel Redundant Point-to-Point Link	I/10-17
	Switched Redundant Point-to-Point Link	I/10-23
	Reactivating a Failed CP (In-Cycle Synchronization)	I/10-26
11	**Technical Specifications: IM 314R / IM 324R**	**I/11-1**
11.1	Technical Specifications of the IM 314R Interface Module	I/11-2
11.2	Technical Specifications of the IM 324R Interface Module	I/11-5
11.3	Readback Delays	I/11-8
12	**Glossary**	**I/12-1**

Introduction to the Installation and Operation of the S5-155H Programmable Controller

1

The S5-155H is a fault-tolerant programmable controller for machine and plant control. It is an event-synchronized master-standby system with a 1-out-of-2 structure. A data link connects line the master to the standby controller.

This introduction is intended to familiarize you with some typical characteristics of the S5-155H, and is aimed especially at those users who are already acquainted with the S5-155H. It has been assumed that the reader is familiar with the S5-155U's functionality.

1.1 Characteristics and Functions of the S5-155H Programmable Controller

S5-155H Performance Characteristics

The S5-155H with CPU 948R is based on the S5-155H with CPU 946R/947R, but its system performance capabilities have been expanded and considerably improved:

- The most important innovations in the S5-155H with CPU 948R are
  - its expandability to as many as 192 redundant analog I/O channels,
  - its fault-tolerant analog outputs,
  - its utilization of the CPU 948's functionality,
  - higher processing speed,
  - page access to one-sided IP/CP permitted,
  - more accurate error locating of configuration errors in the list of the cyclic DB/DX,
  - the range limit for the redundant/one-sided I/O must no longer be configured, i. e. switched, one-sided and redundant I/Os can be mixed.
- Configurable redundancy enables economical solutions (one-, two- or three-channel I/O operation).
- Redundant operation of digital and analog I/O modules.
- NON-STOP operation of redundant I/Os in the S5-155H. The system program supports non-stop operation of redundant I/Os with extensive self-tests for detecting and localizing errors quickly.
- On-line repair of defective I/Os, thus avoiding interruptions in programmable controller operation.
- The support of the COM 155H programmer software, a special program package with its own reference manual, in system configuring and error diagnostics.

Important!

The S5-155H can be operated non-stop.

This means it tolerates the first failure of each **redundant** hardware component. The failed components can be repaired without interrupting operation.

Please note that the S5-155H will fail a) partially or b) completely when the standby component fails before the original fault has been rectified.

a) A partial failure will result, for example, if a redundant input module fails, then its standby also fails before the original module can be repaired.

b) The system will fail completely if, for instance, a central controller fails, then the second central controller also fails before the first can be repaired.

The faster a component is repaired, the lower the risk of further failures!

1.2 S5-155H Applications

Fault-Tolerant Systems

The S5-155H can perform extensive, complex automation tasks while providing a high degree of fault tolerance.

In the majority of cases, fault-tolerant systems continue operation even when one or more faults result in failure of peripheral or central controller components.

Fault-tolerant systems should always be used when it is necessary to keep the probability of a total control system failure (for example a cooling pump control system) to a minimum.

Based on cost, applications for a fault-tolerant programmable controller system can be divided into two categories:

- High production downtime costs per unit of time. Example: Assembly line production.
- High costs even for brief production downtimes. Example: Industrial processes.



Danger

The S5-155H must never be used in plants or installations in which a programmable controller fault or malfunction could result in danger to persons, machines or the environment. Safety-related automation tasks of this nature require the use of a programmable controller which was prototype-tested by the TÜV (German Technical Inspectorate), or the programmable controller must be equipped with suitable interlocks or protective systems which prohibit the occurrence of dangerous operating states.

Fail-Safe Systems

Please note that there is a distinct difference between a fault-tolerant system and a failsafe system.

A fail-safe system also has a redundant component configuration, but enters the STOP mode (in the case of double redundancy) in the event of a fault.

1.3 Redundant S5-155H Configuration

Structure

The S5-155H (central controller) always has a redundant configuration. It consists of two S5-135U/155U central controllers. Three (combinable) I/O redundancy structures are possible:

- One-channel I/O module configuration ("switched");
- Two-channel (1-of-2) I/O module configuration;
- Three-channel (1-of-3) module configuration.

One-Channel Configuration

A one-channel I/O module configuration should be used when the application requires only central controller redundancy (Figure 1-1).

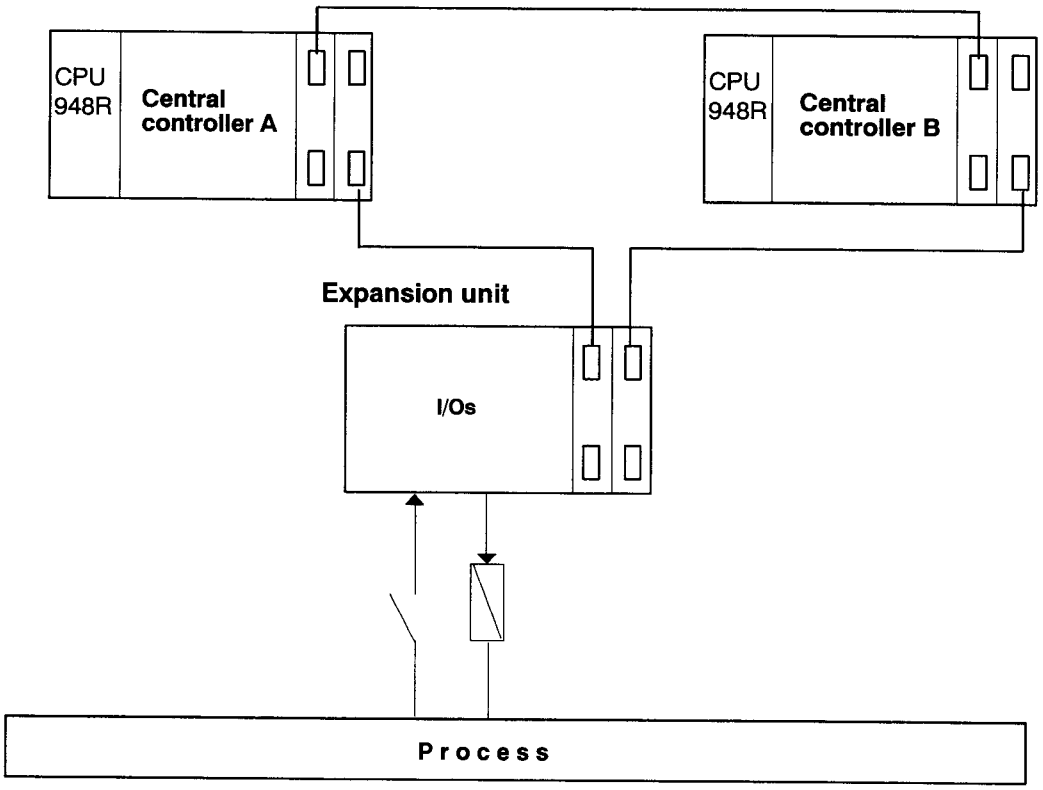


Figure 1-1 Structure of the S5-155H with One-Channel I/O Module Configuration ("Switched Mode")

Multi-Channel Configuration

A two-channel or three-channel configuration should be implemented whenever requirements dictate that input and output modules should also have the highest possible degree of fault tolerance (Figures 1-2 and 1-3).

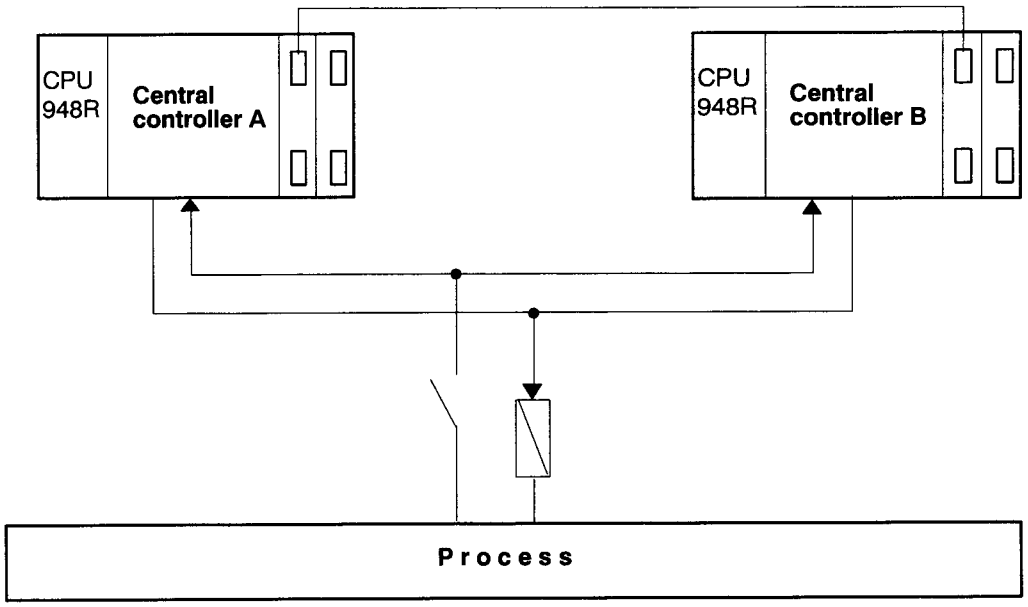


Figure 1-2 Structure of the S5-155H with Two-Channel Redundant I/O Module Configuration

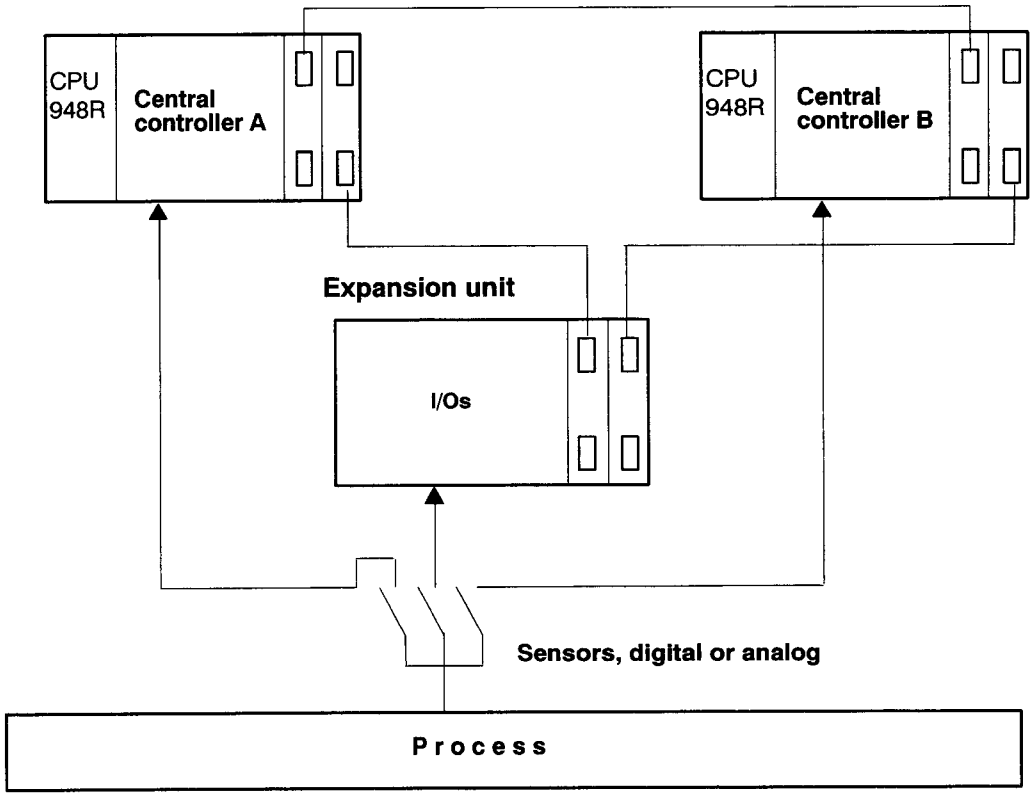


Figure 1-3 Structure of the S5-155H with Three-Channel Redundant Module Configuration

Hybrid Configuration

The three configurations can be combined as needed, thus making it possible to create configurations tailored to meet the fault-tolerance requirements of any given application.

For those parts of a plant which do not require fault tolerance, expansion units (EUs) can be interfaced to each of the central controllers on a 1-out-of-1 basis (as for a one-channel S5-155U).

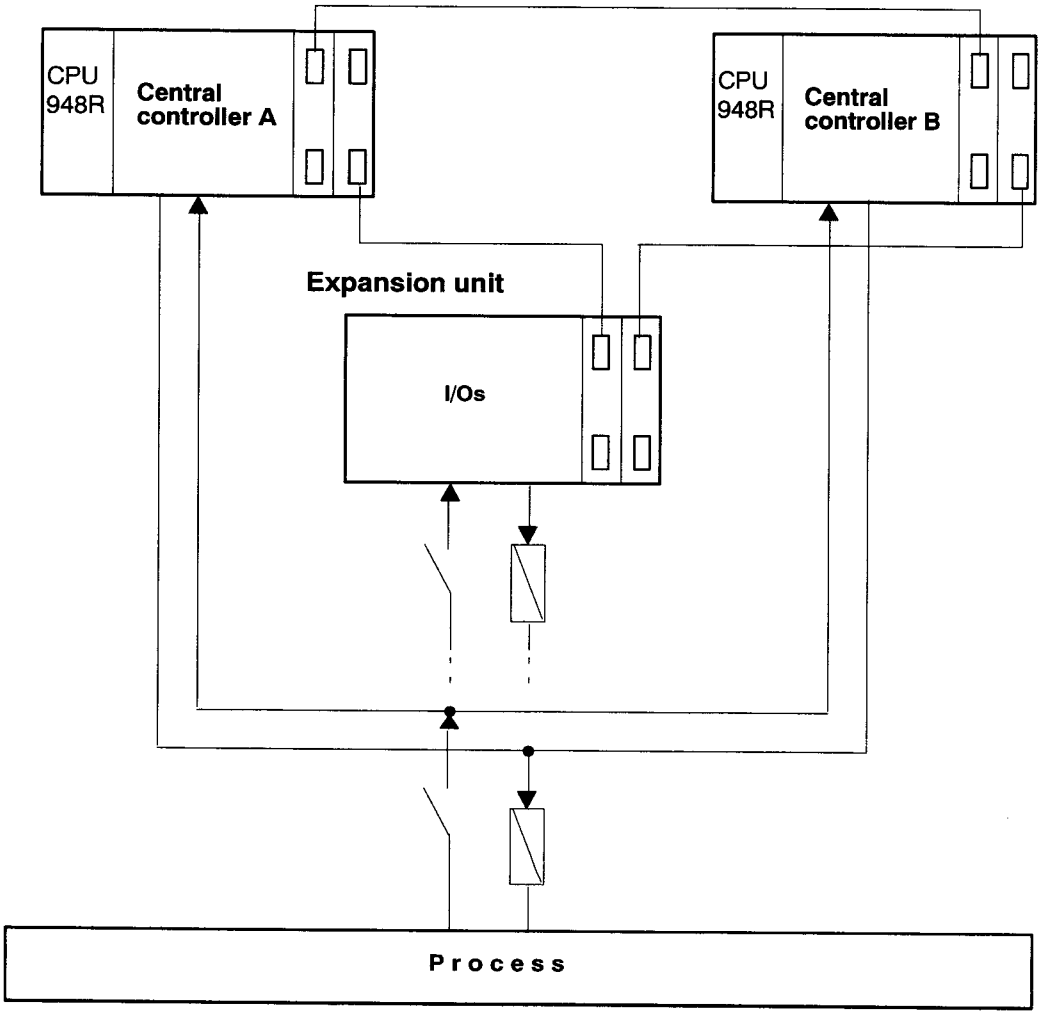


Figure 1-4 Structure of the S5-155U with One-Channel and Two-Channel Redundant I/O Module Configuration (Hybrid Configuration)

1.4 Method of Operation of the S5-155H

S5-155H-Specific Functions

Each of the two central controllers (the master and the standby) contains a CPU 948R central processing unit whose firmware autonomously handles all functions specific to the S5-155H. The most important of these are:

- Event-controlled synchronization of the two central controllers
- Self-tests for memory, processors, central controller link and S5 bus
- Switching from master CC to standby CC
- Error handling and
- Processing of operator entries on the programmer (automatic transfer of data to the other central controller).

The central controller which was powered up and successfully performed its self-test first assumes the role of master CC.

The overview below briefly describes master-standby operation of the S5-155H for one-channel and two-channel configurations.

Operation for **one-channel** ('switched') configuration:

Master CC	Standby CC
• Reads the input signals (PII) • Passes the input signals to the standby CC at the beginning of each cycle • Scans the user program as per the synchronization points • Compares the process output images (PIQs) and • Generates output signals	• Receives the input signals from the master CC • Scans the user program as per the synchronization points • Compares the process output images (PIQs)

PII = Process input image, PIQ = Process output image

Operation for **two-channel** configuration:

Master CC	Standby CC
• Reads the input signals (PII) • Unifies the process input images (PIIs) • Scans the user program as per the synchronization points • Compares the process output images (PIQs) • Generates output signals	• Reads the input signals (PII) • Unifies the process input images (PIIs) • Scans the user program as per the synchronization points • Compares the process output images (PIQs) and • Generates output signals

Operating States and Operating Principles of the S5-155H

The master controller in a fault-tolerant S5-155H programmable controller system can assume the following states while controlling a process:

- **Solo mode**
  The master controller controls the process alone; the standby is inactive.
- **Activation of the standby**
  The master transfers the current data and the current program to the standby.
- **Redundant mode**
  The master controls the process and the standby runs concurrently in an "updated" state, always ready to take over.
- **Error search mode**
  The master controls the process and the standby executes the self-test.

The figure below illustrates the principle of operation, and in particular the interaction between the S5-155H's subunits.

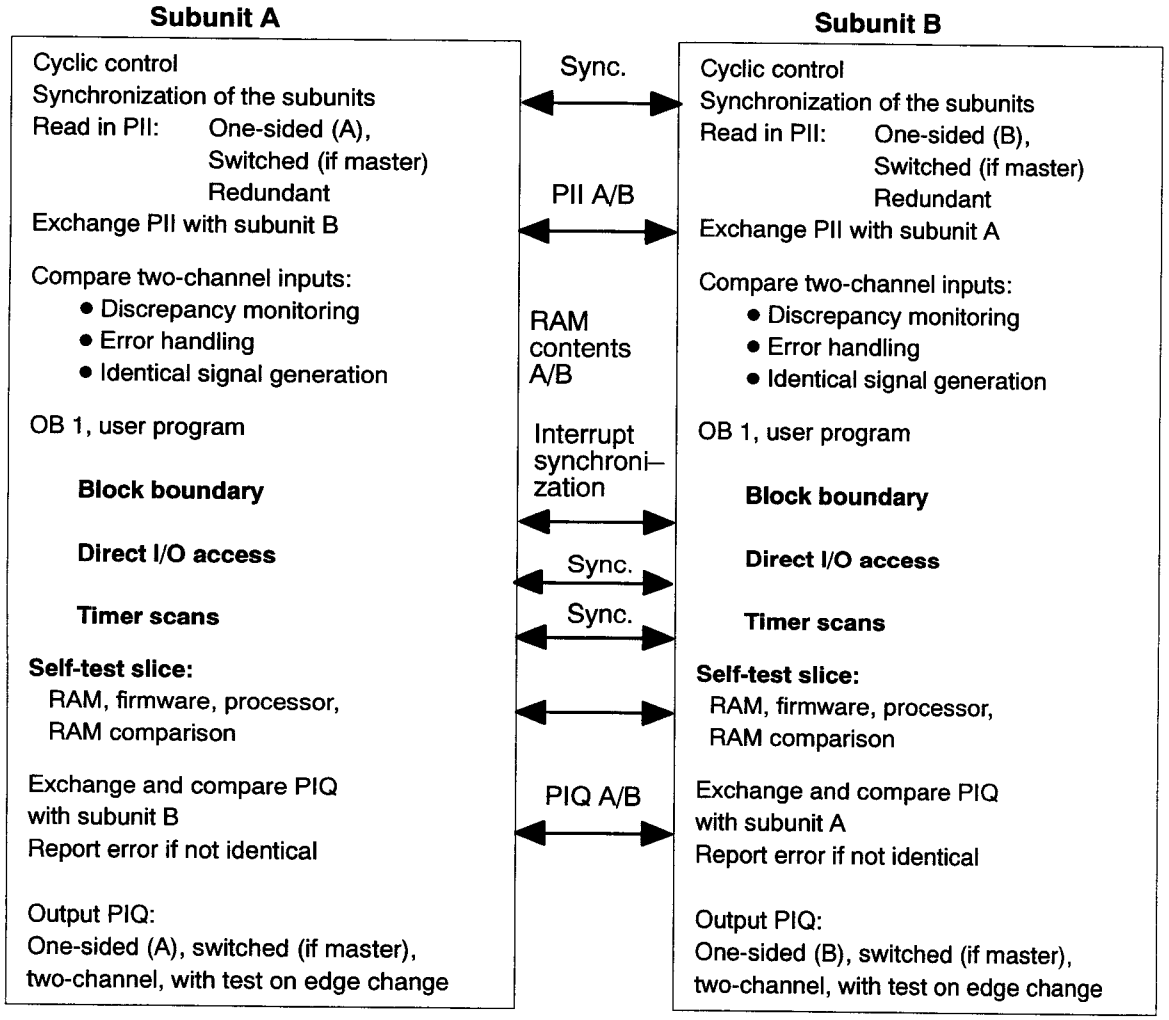


Figure 1-5 S5-155H Principle of Operation

The following is typical of S5-155H operation:

- In a one-channel (switched) configuration, only the master is active.
- In a two-channel configuration, the master and the standby are both active (parallel operation).

The operating system also ensures an ordered functional sequence when using a hybrid configuration (one-channel or multi-channel).

Programming

Essentially, the S5-155H is programmed in the same way as an S5-155U. With the exception of the operations for multi-processor mode, all STEP 5 operations are allowed.

The programmer (PG) is used in exactly the same way as for the S5-155U. In redundant systems, it is connected to only one of the central controllers, and the data entered on it are passed automatically to the second CC.

Program Processing

In addition to normal cyclic scanning of the process images in the subunits' CPUs, additional functions for synchronization and for the interchange and comparison of data and, where applicable, of status information are required during processing of the user program in the S5-155H.

Event-Synchronous User Program Processing

The same user programs run in both subunits (master and standby). Master and standby execute in **event-driven synchronism**. The 155H system program ensures that both subunits work with identical data (see Section 2.3, "Event-Driven Synchronization").

Process Input Image (PII)

At the beginning of each cycle, master and standby read the process images of the one-sided and redundant inputs assigned to them. The process image of the switched inputs is read in only by the master. The two subunits then exchange the entire process input image.

Discrepancy Monitoring

The 155H's system program uses a separate timer for each redundant input. These timers are used to monitor discrepancies. If a specific input signal differs between the master and the standby, this discrepancy is tolerated for the period of time programmed by the user (10 ms to 320 s). These **discrepancy timers** are updated once per cycle, following exchange of the PII. If the two input signals are still not the same when the set time has elapsed, the defective input in the master or standby is located and entered in the error data block.

Self-Test Functions

To detect errors as quickly as possible, the 155H's system program executes self-test functions during restart and cyclic program processing. These functions test the contents and state of memory, processors and I/Os, and make comparisons between the subunits. The functions are processed cyclically in "test slices". The number of test slices may be configured by the user. Please note that the normal scan time of your STEP 5 program is increased by the execution time of the test slices.

Process Output Image (PIQ)

When the user program in OB 1 has been completely processed, the master and the standby exchange and compare **process output images**. If there is a discrepancy, an error is reported. The standby controller enters the "Error search" mode and the master continues in "Solo" mode. The process output images are output to the I/Os.

Scan Cycle Time

The S5-155H has a longer scan time than the S5-155U due to the 155H-specific auxiliary functions. Some of the reasons for the increased scan time are listed below:

- The self-test; the user can configure the required time between 2 ms and 38 ms per cycle.
- The time the 155H operating system needs to process the process images (approx. 15 ms) and
- The time needed by the synchronization commands to process Transfer statements.

  Example: The 'L PW' statement is also called upon to transfer the loaded I/O word to the second CC.

  The time needed to process the Transfer statement depends on the number of synchronization commands in the user program. The typical time needed is approximately 5 % to 15 % of the S5-155U's scan time.

Configuring and Error Diagnostics with COM 155H

The COM 155H software supports both configuring and error diagnostics via the following functions:

- Configuring of H-specific data; this includes
  - specifying which I/O modules are redundant and which are not,
  - specifying the data block in which the system is to report errors (the so-called error DB), and
  - defining the hardware configuration.
- Error diagnostics, i. e., reading out, interpreting and displaying the information found in the error DB
- Making a hardcopy printout to document the configured data
- General system handling tasks.

1.5 S5-155H Hardware Configuration

Minimum Configuration

The fault-tolerant S5-155H programmable controller consists of standard components of the SIMATIC S5 range.

The S5-155H's minimum configuration consists of two S5-135U/155U central controllers (master and standby), each with

- one built-in power supply unit and
- one CPU 948R.

The two central controllers are interconnected over a parallel interface. This interface consists of an IM 304 module (in one central controller), an IM 324R (in the other central controller), and a 721 connecting cable. The parallel interface is used for data exchange between the master and the standby.

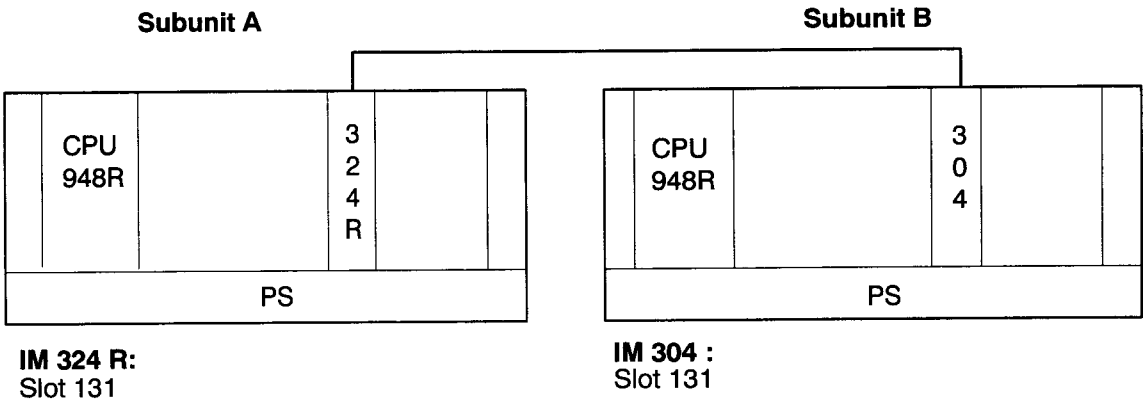


Figure 1-6 Minimum Configuration of the S5-155H

Maximum Configuration

Building on the minimum configuration, the system can be expanded by adding S5 modules:

- Digital and analog input/output modules (I/Os)
- Communications processors (CPs)
- Intelligent I/Os (IPs)

A maximum of sixteen expansion units (EUs) with up to eight I/O buses can be connected to one S5-155H.

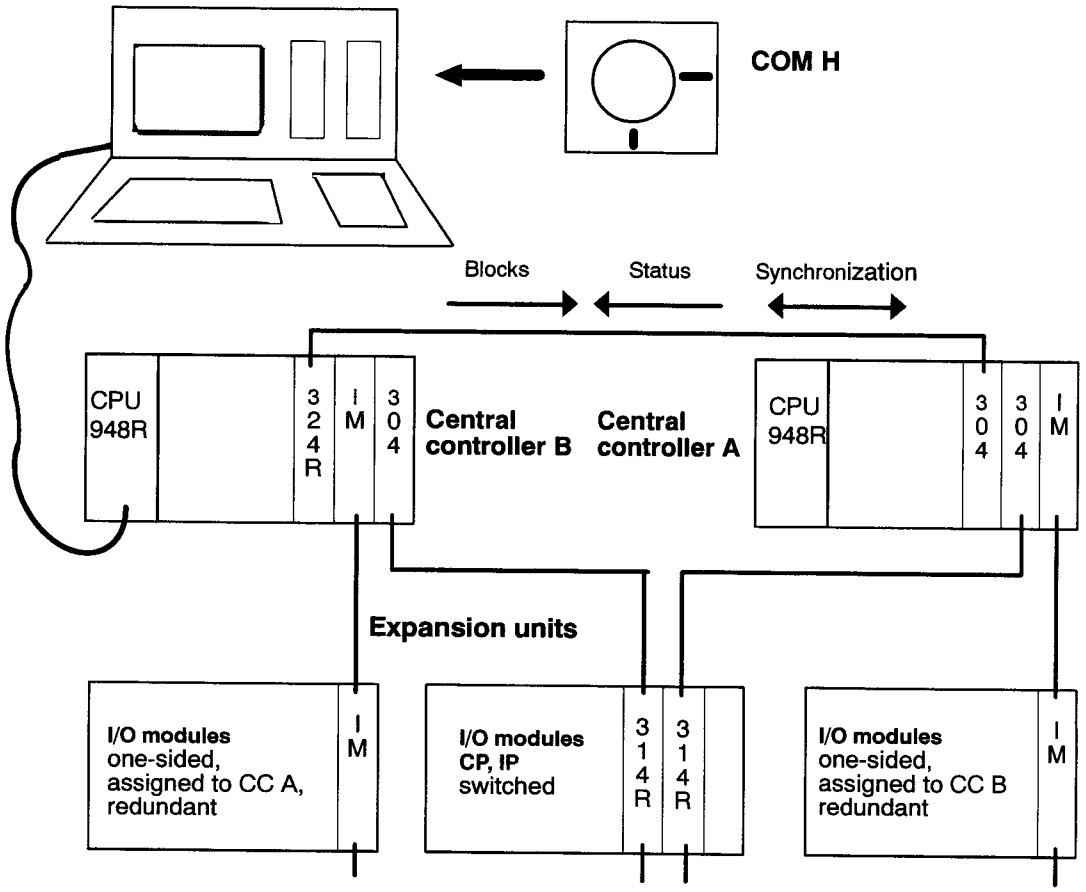


Figure 1-7 Sample S5-155H Configuration, System Structure

1.6 Software

155H System Program

The 155H system program is a modified 155U system program, expanded to include a number of redundancy-specific functions. It is an integral part of the central controller, which means that the whole program memory of the S5-155U is available for the user program.

Important!

The 155H system program reserves

- the data block DX 1
- the error data block (number specified by user)
- the RAM data block (number specified by user)
- the H flag word (number specified by user)

Only those functions are listed here which have been added to the 155**H** but are not part of the 155**U** system program. These and all other functions are described in detail in the Programming Guide for the CPU 948R (Volume 2 of this manual).

The expansions and modifications in the 155H system program affect the following system characteristics and capabilities:

Function	See Chapter/Section
– Mode of operation and status	2.1
– Activating the standby controller	2.2
– Event-driven synchronization	2.3
– Switching from standby to master controller	2.4
– Self-test	2.5
– Error search mode	8.1
– Redundant I/Os	4.2, 4.3, 4.4
– Operation with the IM 314R	11.1
– Time characteristics	7
– Troubleshooting and repairs	9

The various functions are discussed in more detail in the sections indicated.

STEP 5 User Program

In the S5-155U, you can use all STEP 5 user programs that can execute in the S5-155U, provided you observe the following.

Important!

The following applies to the S5-155H:

- The S5-155H can no longer be used in multiprocessor mode.
- The following integral special functions cannot be executed on the S5-155U: OB 126, OB 200, OB 202 to 205, OB 223.
- The SED and SEE operations are only for multiprocessor mode, and therefore not permissible for the S5-155H.

STEP 5 Program

In addition, please note the following when writing your user program for the S5-155H:

- Use only the STEP 5 operations LT and LCT (not LIR, TIR, LDI and TDI) for word-based access to timers.
- The parameters "155U" (or interruptability at instruction boundaries) and "Warm restart" must **not** be selected in DX 0. A DX 0 error resulting in this way causes the CPU to stop.
- You cannot access operating system data words RS 96 to RS 99 via the STEP 5 user program. Instead, use special-function organization block OB 121 or 151 to read or set the date and time; only in this way can you be sure that the same date/time will be entered in both subunits.
- Only the master has direct access to switched I/Os in the restart routine.

Startup

Note the following on startup:

- All DBs/DXs whose contents are changed in interrupt service routines must be entered in the interrupt DB/DX transfer data lists using COM 155H.
- All DBs/DXs whose contents are changed in the cyclic program must also be entered in the cycle DB/DX transfer data list with COM 155H.
- Unusual feature of the S5-155H as regards T PY/T PW and addressing errors:

  If a timeout (QVZ) and an addressing error (ADF) are present when the operation is executed (for unconfigured, non-existent I/Os), then only QVZ is reported in Solo mode whereas both QVZ and ADF are reported in Redundant mode.

Special Functions: OB 124, 125, 254 and 255

An error is reported when one of the following special functions is executed while the master controller is activating the standby: GDB, GXDX, OB 124 "Delete blocks", OB 125 "Generate blocks", OB 254 and OB 255 "Transfer data blocks". Error code "4F": "Function currently impermissible because standby is being activated", is forwarded in accumulator register 1-LL.

Differences between the CPU 948R and the CPU 946R/947R

The CPU 948R is the upwardly-compatible successor to the CPU 946R/947R. The following points should be noted:

- DX1 must be converted with the help of COM 155H V3.0.
- The structure of the error DB has been changed.
- H flag control bit 3 is disabled for the user.

H-Specific System Functions

2

This chapter describes in detail all typical system features and individual S5-155H functions.

2.1 Method of Operation and Operating States of the S5-155H

Operating States

On startup or restart, the S5-155H assumes one of the operating states shown in Figure 2-1:

Solo mode:
The master subunit alone scans the user program and controls the process; the standby subunit is inactive.

Activating the standby:
The master subunit passes the current data to the standby.

Error search mode:
The master subunit scans the user program and controls the process; the standby subunit executes the self-test.

Redundant mode:
The master subunit controls the process; the standby subunit runs in parallel ("updated" mode), and is ready to take over at any time.

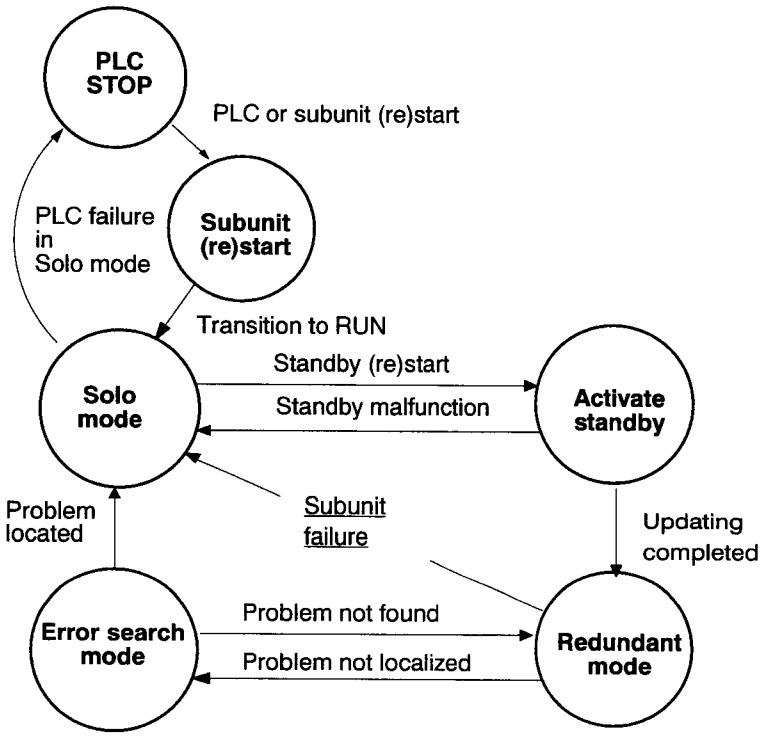


Figure 2-1 S5-155H Modes and Transitions

S5-155H Principle of Operation

In a fault-tolerant programmable controller system, identical user programs run in both subunits. The subunits are "event-synchronized"; that is, only those events which could produce **different** internal "states" in the two central controllers cause the controllers to be synchronized. The internal "state" is determined by the states of the memory areas used (process image, flags, counters, timers and data blocks).

Examples of such 'events' are:

- Direct access operations to one-sided, switched or redundant I/Os
- Timer scans
- Process interrupts and
- Timed interrupts

Processing the User Program

Figure 2-2 shows how an S5-155H with CPU 948R handles I/Os.

a. Each subunit reads the one-sided inputs assigned to it, and the redundant inputs. The 'switched' inputs are read only by the master, which also updates the input image for both subunits and generates a unified input image.

b. One timer location is set aside for each redundant digital input, and is updated by the operating system. The CPU 948R function block sets aside one timer location per redundant analog input which is updated once per FB call. This timer is used as discrepancy watchdog for the redundant digital inputs; that is, the CPU 948R tolerates non-matching input signals for a time period defined by the user (20 ms to 320 s).

c. If a continuous signal discrepancy is ascertained, the defective input is located in subunit A or B by means of an I/O test, entered in the error DB, and passivated.

In order to increase the fault tolerance of the system, errors/faults must be detected and eliminated if possible before the redundant component fails; that is, faults/errors in the standby controller should be detected before the master controller fails. For this purpose, extensive self-test functions for memory, processors and I/Os have been integrated in the CPU 948R.

d. The S5-155H self-test functions run in their entirety every time a subunit is restarted. In cyclic mode, the tests are executed in 'slices' whose number per cycle (scan cycle time increase) can be stipulated by the user.

   During the self-test, as during other time-consuming operating system functions, interrupts are permitted at 2 ms intervals.

e. OB 1, which, like every user program, is synchronized on an event-driven basis, is then invoked.

f. After each program pass, the two output images (PIQs) of the subunits are compared. If they are not identical, an error is reported.

g. The output images are output to the I/O. Also, in the case of two-channel digital outputs, a readback digital input uses a "0" to "1" edge to locate stuck at 0 errors.

The operating system synchronizes and processes system interrupts, process interrupts and timed interrupts at block boundaries. Direct access operations to **switched** I/Os are immediately synchronized and processed.

In the case of one-sided and redundant I/Os, the operating system input signals are also exchanged and compared. Timer scans are immediately synchronized and unified by copying the master's timer value to the standby.

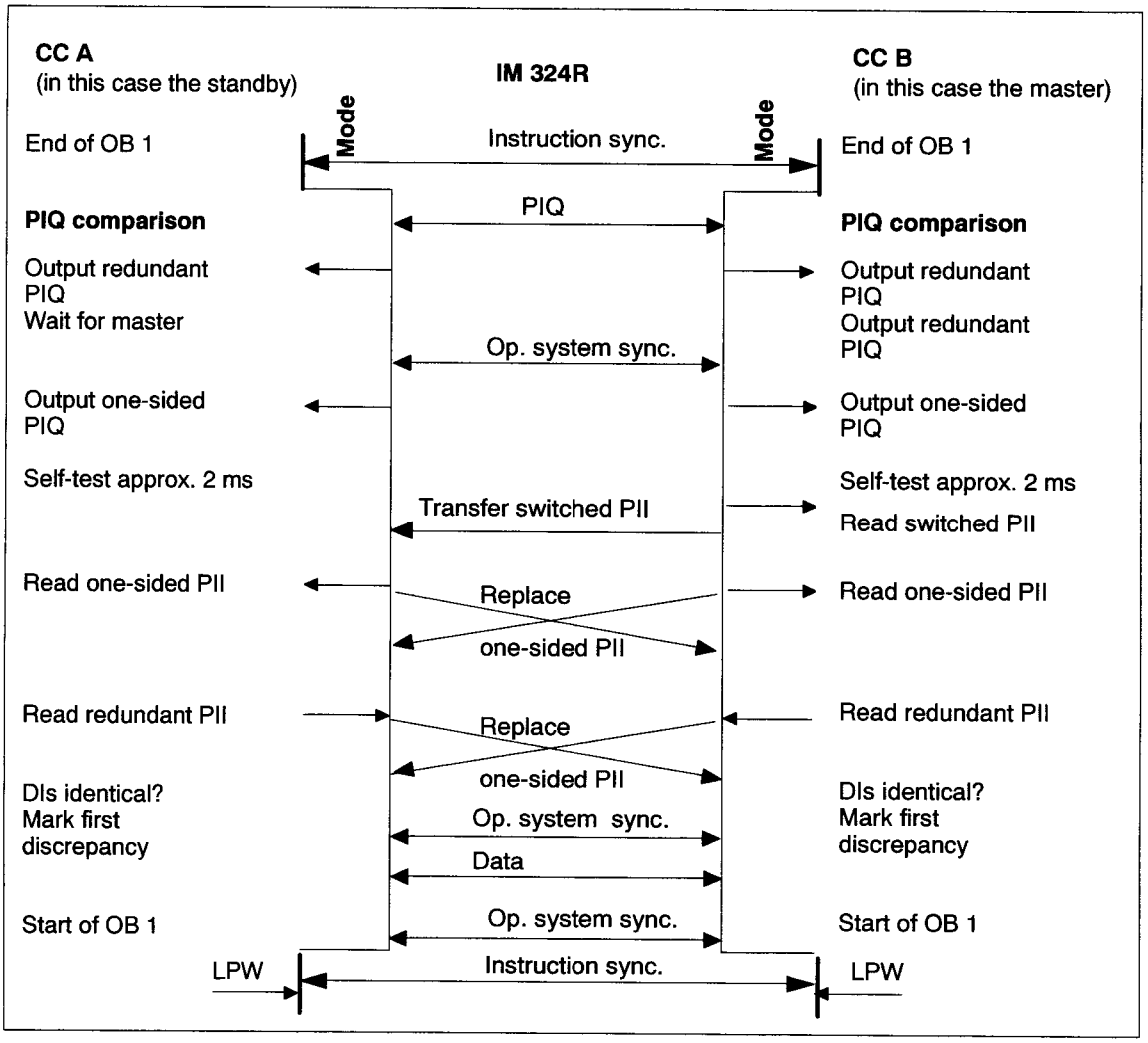


Figure 2-2 Cyclic Process Image Updating of the Inputs and Outputs

Programming the PLC Restart Routine

As regards restarts, be sure to refer to the appropriate chapter in the CPU 948R Programming Guide, (Volume 2 of this manual), where you will find information on all S5-155H restart modes:

- Cold restart: OB 20
- Manual warm restart with memory: OB 21
- Automatic warm restart with memory: OB 22

On-line Function START

Activation of this on-line function does not restart the whole S5-155H system, but only the subunit to which the programmer is connected. The COM 155H function **"RUN SYS"**, in contrast, restarts the whole system.

2.2 Activating the Standby

Activation Process

Activating the standby means **matching the internal states** of the two subunits. After the standby has been enabled in the Restart routine, it sends an "activation request" to the master.

As soon as the request has been made, the two subunits are checked for any discrepancies. This involves checking the following to make sure that there are no differences between the two controllers:

a. The RAM capacity of the standby CPU and the master CPU are the same.
b. The operating system code in master and standby are identical.
c. The checksums of the user code chips are identical.
d. The start addresses of the STEP 5 user blocks are identical.
e. The checksums of the static user data (except for the DB and DX data blocks in cyclic and interrupt-driven program processing) are identical.
f. The checksums of the memory cards in master and standby are identical.

If there is no match in checks a., b. and f., the standby controller reports an error and stops.

If there is no match in tests c. to e., the contents of the master's CPU are copied to the standby's CPU. The activation procedure (the copying of the contents of the master controller's CPU to the standby controller's CPU) is spread over several cycles.

Restart Self-Test and Updating Process

Once the standby controller has been activated (the static data in master and standby are identical), master and standby execute an automatic depassivation routine; this procedure does not erase the error information in the error DB.

The standby controller executes its restart self-test. It then waits to be "updated"; that is, it waits for the arrival of all dynamic data from the master controller.

During the standby activation phase, the red STOP LED and the green RUN LED flash alternately on the standby CPU (at approximately 1/2 second intervals). During the restart test phase, both LEDs show a steady light.

Updating of the standby controller increases the duration of a master controller cycle **on a one-shot basis** by a specifiable amount of time. The updating instant can be chosen on a process-dependent basis; bit position "2" in the H flag word's control byte is reserved specifically for this purpose (see Section 8.5). You can disable the updating procedure by setting this bit, or enable updating by resetting it. This permits you to choose a non-critical process state for the one-shot scan cycle time increase.

Note, however, that disabling updating increases downtimes; that is, it decreases the system's availability or fault tolerance. When the bit is reset to re-enable updating, the enable becomes valid with the next cycle.

Important!

Updating of the standby controller increases the duration of a master controller scan cycle on a one-shot basis by the amount of time specified in the configuration data. To minimize the scan time increase as much as possible, enter only those DB and DX data blocks during the configuring phase with COM 155H which will be modified in the user program (for example DBs in OB 1), and which must therefore be transferred within one cycle when activating the standby.

Since timed and process interrupts are not disabled when activating the standby, also specify the numbers of those data blocks modified in interrupt service routines (such as DBs in OB 13).

Updating the Standby Controller

The controller is updated as follows. The 155H system program

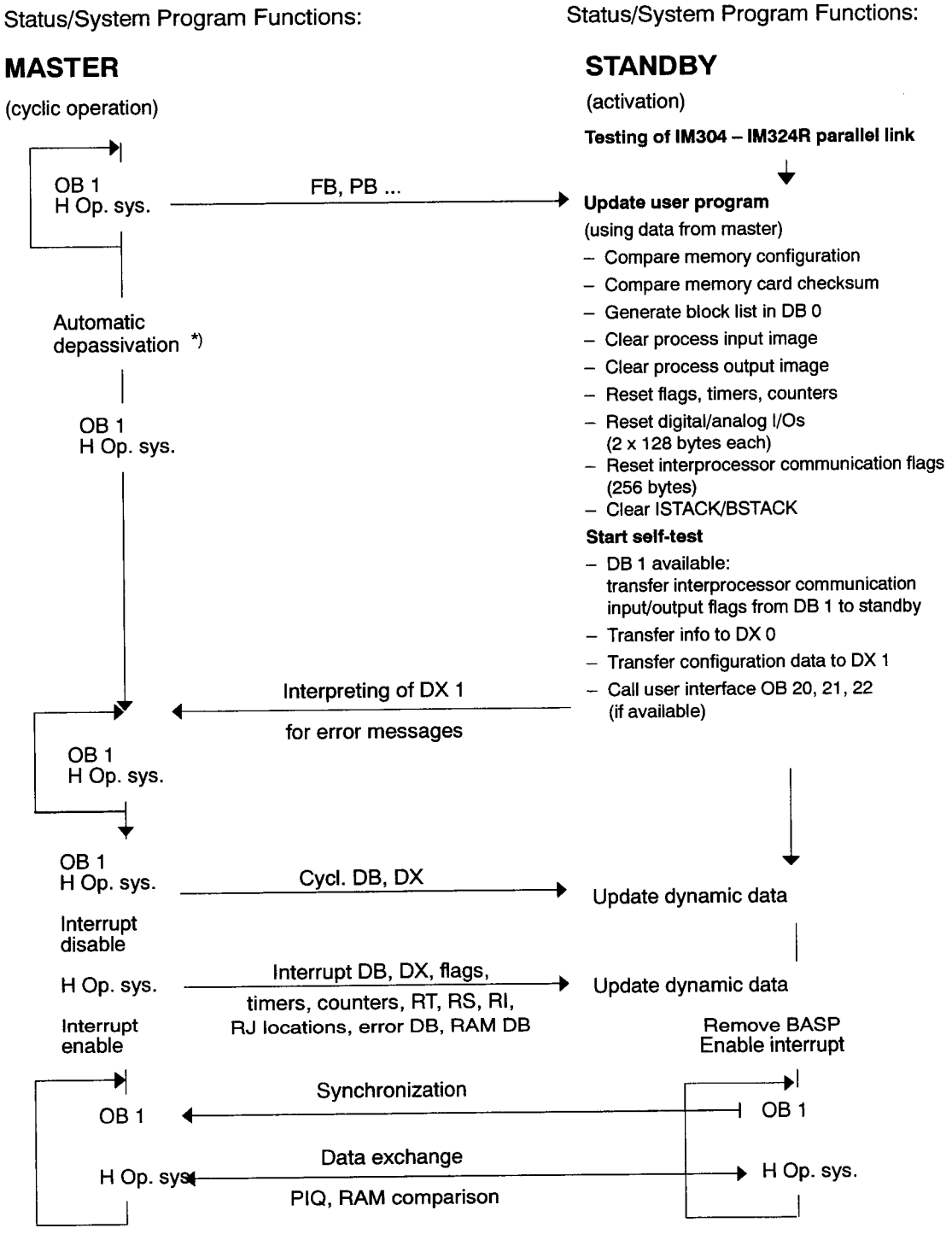
- transfers all configured "cycle DBs/DXs", [1)]
- disables all interrupts,
- transfers all configured interrupt DBs/DXs, [1)]
- transfers all flags, counters, timers, RS, RT, RI and RJ locations, the error DB, and the RAM DB,
- switches to "Redundant mode" and
- enables the interrupts.

[1)] Data blocks processed both in the cyclic program and in an interrupt OB need be listed only once under "interrupt DBs/DXs".

With regard to the above, also please refer to Section 3.3 in Part III of this Volume entitled "Initializing the Activation of the Standby".

Once the standby has been activated and updating completed, both subunits enter the event-synchronized cyclic mode.

The following overview (Figure 2-3) summarizes the activities of the master and standby CPUs during the activation and updating process.



*) Static error image is reset. The error messages in the error DB remain unchanged. The operating system re-enters any unrectified errors in the error DB if those same errors re-occur.

Figure 2-3 Standby Activation and Updating Sequence

2.3 Event-Driven SynchronizationSynch

Switchover from Master to Standby

Both subunits are synchronized to ensure bumpless switching between master and standby at all times.

The synchronization procedure used in the S5-155H is known as "event-driven synchronization", which means that the subunits are synchronized whenever an event occurs which could result in the two subunits having different internal states; for example different process images, flags, timers or communications data. These events include:

- Direct access to I/Os
- Timer scans
- System interrupts
- Process interrupts
- Timed interrupts

Synchronization and Program Processing

The operating system performs subunit synchronization in order to ensure complete transparency of the user program. The user is not aware of this. This means that you can write your program in the same way as for an S5-155U in single-processor mode. The only difference is that the synchronization procedure increases the normal S5-155U execution times of the STEP 5 operations used for direct I/O access, timer scans and block changes (refer to the STEP 5 List of Operations). The execution times of the other operations are not affected.

Synchronization and Interrupt Servicing

In the S5-155H, block-granular interruptibility is the only permissible mode. This means that interrupts are serviced only between blocks. In order to prevent different "internal states" from occurring, entry into the interrupt OB is made at the same place each time, namely, at a "synchronization point". In the CPU 948R, the synchronization point for interrupts is always the next block change. Input byte IB 0 can be used for process interrupts.

Synchronization and System Monitoring

In each subunit, a check is made at every synchronization point to make sure the other subunit is functioning. Depending on the result of the check, a switchover from the standby to the master controller is performed and the message "Standby failure" output.

The synchronization procedure is timed at each synchronization point. The operating system sets the watchdog timer to 30 ms. A check is also made at each synchronization point to make sure that both subunits are executing the same statement (comparison of OP codes). If they are not, the standby controller stops and a "synchronization error" is reported.

Process Image Updating

At the end of OB 1, the entire process output image (PIQ) in both CPU 948Rs is immediately compared and the non-passivated PIQ (redundant PIQ, switched PIQ, one-sided PIQ) output. The self-test is then run.

Following the self-test, the process input image (PII) is read in (redundant PII, switched PII, one-sided PII). The PII is exchanged and unified. OB1 is then re-called.

The required functions are described in detail in Chapter 4.

Automatic Resetting of the Standby Controller's Clock

Due to the inaccuracy of quartz timing, CMOS clocks of the two subunits may begin to show a discrepancy after a while (at a rate of approximately 1 second per day), which can cause problems when transferring from the standby controller to the master. The error codes entered in the error DB show the exact sequence of occurrences, but the time stamp could prove misleading.

For this reason, the two subunit clocks are compared cyclically, and the operating system automatically corrects the standby controller's clock when the discrepancy exceeds 0.05 seconds.

Cycle Errors

If a ZYK error occurs in Redundant mode, the standby controller always stops, while at the same time the scan cycle time in the master controller is retriggered. The subunit which continues its operation in Solo mode functions like an S5-155U; that is, the response to a subsequent ZYK error depends on OB 26.

2.4 Switchover from Standby to Master Controller

Switching Criteria

The synchronization procedure ensures bumpless switching from standby to master controller at all times. This means:

- Transfer does not affect process output signals.
- There is no loss of information in communications with CPs/IPs.
- User program processing is not affected.

Switchover from standby to master takes place in the following instances:

a. Failure of the master CC (BASP, NAU or STOP switch);

b. Initial error search of both subunit CPUs unsuccessful (see "Error search mode")

c. First failure of a master controller's IM 314R when the standby controller has access to a larger number of IM 314R interface modules than the master.

d. First failure of a master controller's I/O bus (wire break, for instance) or failure of an IM 304 when the standby controller has access to a larger number of IM 314R interface modules than the master.

e. First failure of a switched I/O module

f. User issues a software request (H flag control byte) for transfer from standby to master controller.

g. If more than 30 timeouts occur in redundant I/O bytes in the subunit of the master within one PLC cycle.

In cases c. to g., the new standby CPU does not stop, but continues functioning as standby controller.

Functional Sequence of a Switchover from Standby to Master

The standby controller checks the operational status of the master controller at each synchronization point. Failure of a master controller is detected at the hardware level by evaluating the S5 bus signals BASP and NAU in the IM 324R parallel interface module. The standby CPU's operating system detects the failure of the master controller at the next synchronization point, and branches to a routine which executes the following functions:

- Switches over the I/O buses of all IM 314R modules;
- Switches all two-channel I/Os to single-channel operation;
- Switches the operating system to Solo mode; that is, no subunit synchronization;
- If the synchronization point is a direct I/O access operation, that operation will be retried.

Following a standby-to-master switchover, the S5-155H with CPU 948R operates in Solo mode. The H error system OB is invoked; you can program the desired response in this OB.

The following occurs with the I/Os allocated to the failed subunit:

- PIQ and PII are set to zero
- No 'ADF' error is reported when this PIQ/PII is accessed
- Timeout is reported in the event of direct access to one of these I/Os.

2.5 Self-Test

Self-Test and Fault Tolerance

Essentially, the S5-155H achieves its high degree of fault tolerance through its multi-channel capability coupled with speedy repair following detection of a malfunction. If a controller is considered to be fault tolerant only when its status is that of "no malfunction", it follows that the fault tolerance of an H system is increased further by its self-test (Figure 2-4).

The diagram and the formula below show that a self-test increases the fault tolerance of an automation system. The amount of time in which the controller malfunctions is reduced to a minimum.

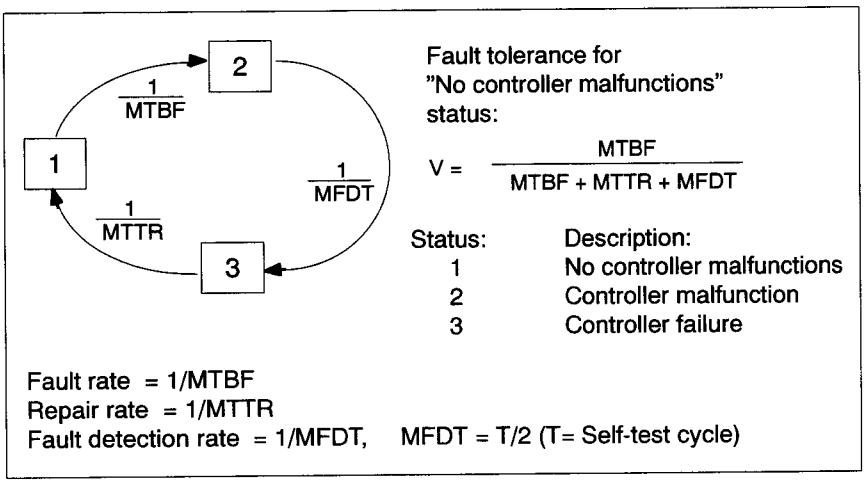


Figure 2-4 Importance of the Self-Test to Fault Tolerance

In contrast to F systems, an H system's highest priority is not fault detection, but rather fault localization. This is required in order to control the fault. The S5-155H's self-test routines run in both CPUs. They detect and localize hardware failures in a minimal amount of time and with little programming effort. To localize a fault, it is necessary only to find out which modules are faulty and to replace them.

Self-Test Strategy

Which self-test routines execute depends on the S5-155H's operating status.

- **Self-test in the Restart routine**

  The entire self-test is run when a central controller is restarted. If a fault is detected at this stage, the CPU stops. An error message is entered in the error data block. Because the self-test takes longer than one minute, it can be skipped on a warm restart of the master controller (see "H Flag Control Byte").

  During execution of the self-test in the Restart routine, the RUN and STOP LEDs on the frontplate show a steady light. A complete self-test is run in all Restart modes.

- **Self-test in cyclic mode**

  Each time OB 1 is processed (once per cycle), part of the self-test is executed **in small slices** (2 ms test slices). The self-test therefore runs, transparent for the other software, in the background until a hardware failure is detected.

  You can configure the execution time for the self-test in increments of 2 ms. If you define a value which exceeds a test slice, the occurrence of a timed or process interrupt is queried after each test slice and the interrupt serviced **before** the next test slice. The execution time for the self-test increases in this case by the time needed to service the interrupt.

  A value of between 1 and 19 may be defined as the number of test slices to be executed per PLC cycle (refer to Chapter 3 in Part III of this Volume, entitled 'Configuring and Initializing'). These values correspond to self-test execution times of from 2 to 38 ms.

Tested System Components

In the S5-155H, such important system components as CPU, memory, I/Os and communications links are continually tested and monitored.

- CPU test

  Includes testing of STEP 5 operations, timers, CMOS clock, interrupt mask and the scan time monitor.

- Firmware/RAM test

  Comparison of the RAM in both subunits and a checksum test of the OB, SB, PB, FB and FX blocks.

- I/O bus test with IM 314R

  Tests for short-circuits and breaks in the 721 I/O bus cable to the IM 314R.

- I/O bus test for page addressing

  Page addressing is tested in cyclic mode once in every complete test run. The test detects the following faults:

  - A CP/IP reacts to (acknowledges) not only its own interface number, but also to the other 255 interface numbers.
  - An acknowledgement is issued by an unassigned interface number. All unassigned interface numbers are tested. The test is also executed once on each restart.

- IM 304R/IM 324R parallel interface test

  The parallel interface's dual-port RAM is tested on both subunits A and B. The purpose of this test is to locate any short-circuits or cable/wire breaks.

Locating Facilities (LF) Test

The user must configure one additional digital input and one additional digital output for each redundant digital input or output byte for which error recognition as well as error localization is to be implemented. Since these are specifically for the purpose of error localization, they are referred to as locating digital inputs (L-DIs) and locating digital outputs (L-DQs) (see Figures 4-4 and 4-8/4-9).

The L-DIs and L-DQs configured for each redundant DI or DQ are referred to collectively as the locating facility (LF). The LFs are tested once every 10 hours.

The locating facilities for the redundant DIs are tested for "stuck at 0" errors and (on restart only) for "stuck at 1" errors. The locating facilities for the redundant DQs are also tested every 10 hours for "stuck at 0" and "stuck at 1".

Self-Test in Error Search Mode

When an error is detected which, because it was found during a RAM comparison test (only valid blocks are compared), could not be allocated to a specific subunit, the standby controller enters the Error Search mode. This mode is also entered when a discrepancy is found when comparing the output images of the two subunits. In Error Search mode, the self-test is not executed in slices, but rather **in its entirety**, which takes approximately 10 to 30 seconds.

Functional Sequence in Error Search Mode

Example:

The two subunits exchange and compare the entire process output image (PIQ) at the end of each cycle. If a discrepancy is found during this cyclic PIQ comparison test, the memory locations in the PIQ are tested for "stuck at 0" and "stuck at 1" errors. If such an error is found, only the malfunctioning subunit stops.

If the error cannot be localized, the 155H responds as per user specifications (see Part III of this Volume, Section 3.2, 'Initializing the Operating System'). If the configured response is "0", the standby controller enters the "Error Search" mode. The master controller runs in "Solo" mode.

If the self-test locates a fault in the standby controller, the standby stops. Otherwise, the standby controller is activated and a standby-master switchover initiated. The PLC now runs in Redundant mode. Should another comparison error be found, the new standby controller enters the "Error Search" mode while the new master continues in "Solo" mode.

If the self-test locates an error in the new standby controller, which is now in Error Search mode, the standby controller stops and reports an error. If the self-test cannot locate an error in this subunit either, the subunit stops with "Non-locatable error" if the second comparison error occurred within the same test cycle.

CPU 948R

3

This chapter contains the hardware description and technical specifications for the CPU 948R central processing unit. In addition to comments regarding application, you will find all information and data needed for installing the module and putting it into operation. This includes information on inserting and removing the module as well as on control elements and indicators on the module's frontplate.

Details on programming can be found in the Programming Guide for the CPU 948R (Volume 2 of this manual).

You can insert a memory card containing your user program into the CPU 948R.

3.1 Technical Specifications of the CPU 948R

Application

The CPU 948R can be used in S5-135U/155H central controllers. Multiprocessor configurations are not possible in fault-tolerant systems.

The CPU 948R is available in two versions:

- The CPU 948R-1 with 640 Kbytes of internal user memory (RAM).
- The CPU 948R-2 with 1664 Kbytes of internal user memory (RAM).

A SIMATIC S5 flash EPROM memory card (referred to in the following simply as memory card) can be plugged into the CPU 948R as storage medium for the user program and user data. On an overall reset, the contents of the memory card are copied to the CPU's internal RAM.

The CPU 948R is programmed in STEP 5 (LAD, CSF, STL, SCL). The CPU 948R processes all STEP 5 operations at a very high speed, and is also equipped with a high-speed floating-point arithmetic facility.

The following program processing levels are possible:

- Cyclic
- Time-controlled (9 different time grids, clock-controlled, delayed interrupt)
- Interrupt-driven over the S5 bus (eight process interrupts at block boundaries via IB0)
- 'Soft STOP'

Configuration

The CPU 948R's electronics, including RAM, are on two PCBs. These PCBs have Eurocard format, and are bolted to one another. They must never be separated.

The module's frontplate has a width of 2 2/3 standard slots, or 40 mm. The CPU 948R takes up two slots in the central controller rack.

3.2 CPU 948R Installation and Startup Procedures

Note

All jumpers on the module are required for the manufacturer's quality inspection. The jumper configuration must not be changed in any way.

Removing and Inserting the Module



Caution

Always switch off the power before removing or inserting the module.

The CPU 948R's basic board and its expansion board are a unit, and must not be separated.

Inserting the module

Proceed as follows to insert the CPU in the central controller:

Step	Procedure
1	Loosen the upper locking bar on the central controller and check to make sure that the module's locking bolt is correctly positioned (that is, with the slit in the vertical position).
2	Select the right slot (check the label on the locking bar). Insert the CPU (slot 11), aligning it to the left.
3	Push the module evenly onto the guide rail until the lever above the locking bolt is in a vertical position.
4	Press in the locking bolt on the underside of the module and turn it 90° to the right.
5	Affix the upper locking bar.

Removing the module

Proceed as follows to remove the CPU:

Step	Procedure
1	Loosen the upper locking bar on the central controller.
2	Loosen the module's locking bolt.
3	Press down on the lever and remove the module by pulling it forward and out of the controller.

Control Elements and LEDs

The control elements and LEDs are on the CPU module's frontplate (Figure 3-2).

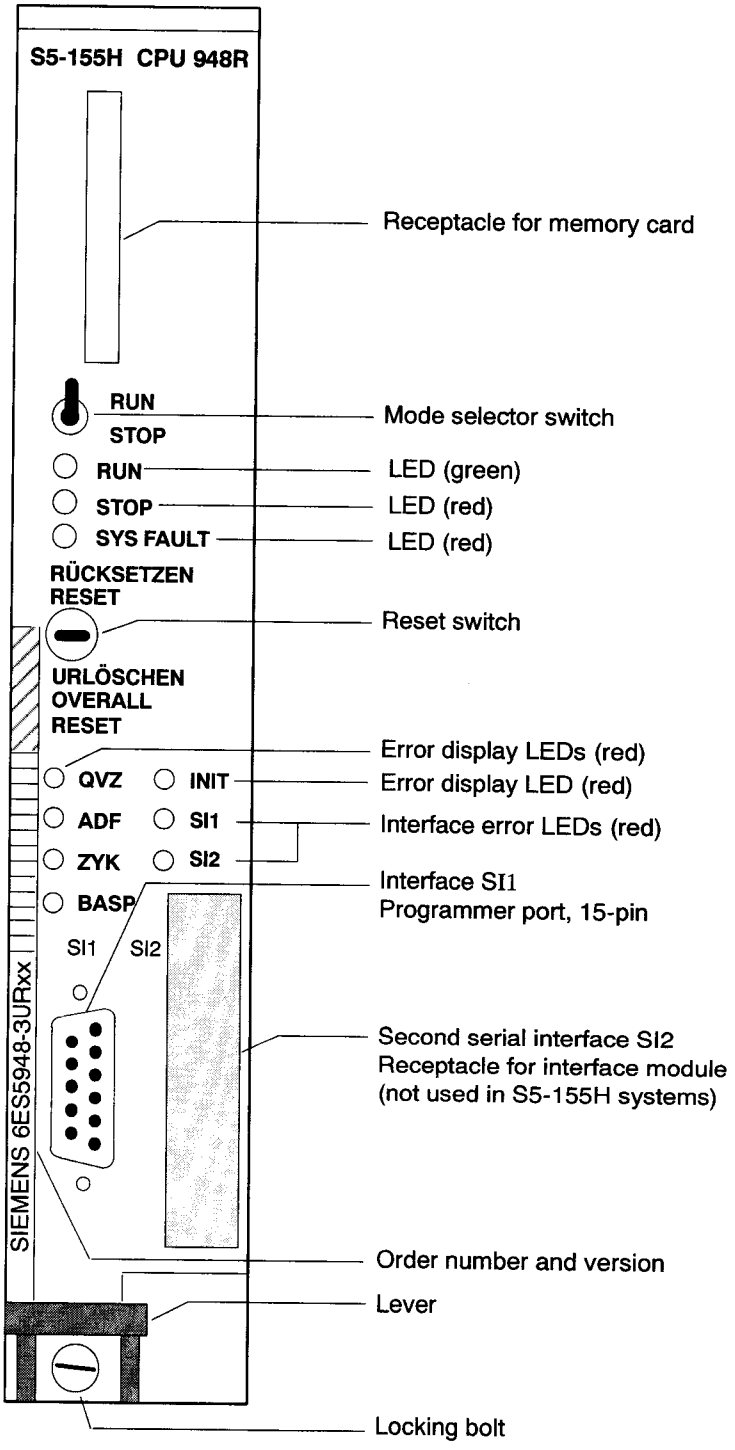


Figure 3-1 CPU 948R Control Elements and LEDs

Mode Selector Switch

The mode selector switch has two positions:

- RUN
  When the mode selector switch is set to 'RUN' and the green LED is on, the CPU 948R is processing the user program.
- STOP
  The CPU 948R goes to 'soft STOP' when the user switches from 'RUN' to 'STOP'. The red "STOP" LED goes on.

Reset Switch

The restart functions "Overall reset", "Cold restart" and "Cold restart with memory" can be initiated via the mode selector switch and the reset switch:

Function	Switch position	Description
Overall reset (OVERALL RESET)	Down	An overall reset reinitializes the internal RAM (meaning any data currently in RAM are erased and the contents of the memory card, if any, are copied to internal RAM). A complete self-test is then executed.
Cold restart (RESET)	Up	A cold restart, or RESET, resets all flags, timers, counters and the process image. OB 20 is then called, and user program processing begins again.
Cold restart with memory	Middle	User program processing begins again, but all flags, timers, counters and the process image retain their current states.

Status LEDs

The following overview describes the LEDs for "RUN", "STOP" and "SYS FAULT".

The "STOP" LED signals a 'soft STOP', the "SYS FAULT" LED a 'hard STOP'.

In 'soft STOP' mode, the CPU 948R can scan the user program (OB 39) cyclically, but the digital outputs remain disabled. In 'hard STOP' mode, no program can run; the CPU is at a 'standstill'. This state can be exited only by switching the power off and then on again.

LED			Mode
RUN	STOP	SYSFAULT	
RUN or RESTART mode			
On	Off	Off	The CPU is in RUN mode and is master (cyclic operation).
Flashes	Off	Off	The CPU is in RUN mode and is the standby (cyclic operation).
On	Flashes	Off	The CPU is master; the parallel link has failed.
On	On	On	Appears briefly when the controller is switched on.
On	On	Off	The CPU executes the self-test on startup.
Off	Off	Off	The CPU is in the RESTART mode or PROGRAM TEST.
'Soft STOP' mode			
Off	On	Off	The CPU is in 'soft STOP' mode. After switching on the power when the mode selector switch is at STOP and no errors occurred during initialization. A restart is possible.
Off	Flashes at high frequency	Off	The CPU is in 'soft STOP' mode. An overall reset was requested via mode selector switch/reset switch or by the operating system. A restart is possible only after performing an overall reset or after eliminating the problem and then performing an overall reset.
Off	Flashes at low frequency	Off	The CPU is in 'soft STOP' mode. • An error was detected during cyclic program processing. The CPU is at STOP because no appropriate error handling routine was programmed. When you move the mode selector switch from RUN to STOP, the LED will once again show a steady light as long as the error does not re-occur. • When error conditions exist, for example, selection of an illegal Restart mode, DB1/DX0 errors, and so on. • When a STOP operation (STP or STS) was encountered in the user program. • In the event of a PROGRAM TEST programmer function for this CPU. • Some programming errors and controller faults also set the 'ADF', 'QVZ' or 'ZYK' LEDs.
'Hard STOP' mode			
Off	Off	On	The CPU is in 'hard STOP' mode. • When error-free execution of the system program is no longer possible, the CPU enters the 'hard STOP' mode. • Reasons for a 'hard STOP': – Timeout (QVZ) or parity error (PARE) in system RAM – ISTACK overflow – STEP 5 operation "STW" • A 'hard STOP' can be exited only by turning off the controller and switching it on again.
'Activate standby' mode			
Flashes	Flashes	Off	The CPU is the standby, and the mode is 'Activate standby'.

Error LEDs

The following overview shows why and when each error LED goes on:

LED	Description
QVZ (timeout)	A module addressed by the program no longer acknowledges, although/because: • it either acknowledged in the process image (IB 0 to 127, QB 0 to 127) on a CPU 948R cold restart and was entered as available in the so-called '9th track' • or was entered in DB 1 (address list) and was recognized as being available on a cold restart • or was addressed in direct access mode • or no access is possible to the data handling blocks on the module. Possible causes: • Module failure; failure of the expansion unit • Module was removed during operation, while the CPU was at STOP or when the controller was switched off and there was no subsequent cold restart. • Failure of the enable voltage L+
	A timeout occurred while user memory was being accessed.
ADF On	The user program referenced an address in the process image which was not entered in DX 1.
ZYK On	The watchdog timer used for scan time monitoring responded and cyclic program processing was interrupted.
BASP On	Command output is disabled; the digital outputs are set directly to the safe state (0).
INIT On	This LED briefly shows a steady light during the initialization procedure which follows POWER ON, and during operation in the event of a system fault.

A detailed description of interrupt servicing and error handling procedures can be found in the Programming Guide for the CPU 948R (Volume 2 of this manual).

Interface Error LEDs SI1 and SI2

Reasons why the interface error LEDs go on:
LED SI2 is always off unless there is a CPU fault.

LED SI1	Reason
On	No communication possible; internal error.
Off	Both interfaces are initialized and ready.

Startup

The CPU must be inserted into the correct slot in the central controller. The backup battery must be inserted and in good working order before the CPU can be put into operation.

Overall Reset

Proceed as follows:

Step	Entry/Activity	Result
1.	Set the mode selector switch to STOP.	
2.	Switch on the mains power.	The following LEDs on the CPU must go on: • Red "STOP" LED (flashes at high frequency) • Red "INIT" LED (briefly) • Red "BASP" LED
3.	Hold the reset switch in the 'OVERALL RESET' position and move the mode selector switch from 'STOP' to 'RUN'.	The red "STOP" LED and the RUN LED now show a steady light.

The red 'SYS FAULT' LED also goes on only if an error occurred during the overall reset. In this case, repeat the measures described. If necessary, switch the power off and then on again. If the LED is still on, the CPU module is defective.

Cold Restart

Continue as follows:

Step	Entry/Activity	**Result** for master
4.	Set the mode selector switch to 'STOP'.	
5.	Hold the reset switch in the RESET position and move the mode selector switch from 'STOP' to 'RUN'.	1. The red 'STOP' LED goes out 2. The red 'STOP' LED and the green 'RUN' LED go on 3. The green 'RUN' LED goes on 4. The red 'BASP' LED goes out The CPU is not in 'RUN' mode, but is still without a user program.

Cold Restart with Memory Function

The reset switch allows you to execute a manual cold restart (RESET) with memory. Please refer to the Programming Guide for the CPU 948R (Volume 2 of this manual) to see when a manual cold restart (RESET) with memory is permissible.

Step	Entry/Activity	**Result** for master
1.	Move the mode selector switch from 'STOP' to 'RUN'.	1. The red 'STOP' LED goes out 2. The red 'STOP' LED and the green 'RUN' LED go on 3. The green 'RUN' LED goes on 4. The red 'BASP' LED goes out

For maintenance purposes or in the event of an error, the above-mentioned startup procedures, without a user program, can be used to see whether the CPU is functioning properly.

CPU 948R Interfaces

This chapter provides information on CPU 948R interfaces (ports) which can be used to connect a programmer or personal computer.

- **Programmer port SI1**

You can connect a programmer to this frontplate port regardless of the CPU's current mode.

- **Interfacing with SINEC H1 over the parallel wiring backplane**

Connecting a programmable controller with a programmer via SINEC H1 enables very-high-performance communications between the two. For example, the loading of application software into the CPU while in STOP mode is up to eight times faster than it would be if a serial port were used.

In addition to a CPU 948R, you also need a CP 143 (revision level 3), a PG 7xx programmer with SINEC H1 interface, and the STEP 5 "Single Tasking" (version 6.3 or newer) or "Multi-Tasking" (version 6.0 or newer) software.

Note
You cannot interface a subunit to SINEC H1 and still use the serial port.

Interfacing to SINEC H1 is discussed in detail in the Programming Guide for the CPU 948R.

Specifications

Important for use in the USA and Canada

The following approval certifications have been granted:

UL Listing Mark
Underwriters Laboratories (UL) to
Standard UL 508, Report E 85972

CSA Certification Mark
Canadian Standard Association (CSA) to
Standard C 22.2 No. 142, Report LR 63533

Technical Specifications

Characteristic/Function	Value
Degree of protection	IP 00
Ambient conditions	
Ambient temperature during operation	0 to 55 °C
Temperature change during operation	Max. 10 K/h
during transport and while in storage	Max. 20 K/h
Relative humidity	Max. 95 % at 25 °C, no condensation
Pollutant emissions: SO_2	10 cm^3/m^3, 4 days
H_2S	1 cm^3/m^3, 4 days
Oscillation during operation	10 ...58 Hz (const. amplitude 0.075 mm) 58 ...500 Hz (const. acceleration 1 g)
Noise immunity, electromagnetic compatibility (EMC)	See Technical Specifications for the S5-135U/155U
Interference suppression Limit class	A (to VDE 0871)
Conducted interference on AC supply lines	2 kV (to IEC 801-4 (Burst) 1 kV (to IEC 801-5) Line to line 2 kV (to IEC 801-5) Line to ground
Immunity against static discharge to IEC 801-2 (ESD)	Proper installation ensures an interference immunity of 4 kV contact discharge (8 kV atmospheric discharge).
Immunity against high-frequency radiation	HF current-sourcing to IEC 801-6 Limit class 3 (up to 200 MHz), corresponding to 3 V/m
Auxiliary power:	
Supply voltage	5 V ± 5 %
Power consumption at 5 V	3.6 A typ.
Backup voltage:	3.4 V
Backup current:	10 µA (at 25 °C) typ.

	P area	O area	IM3 area	IM4 area	Total
Digital inputs with process image	max 1024	–	–	–	max. 1024
Digital inputs without process image	max. 1024	max. 2048	max. 2048	max. 2048	max 7168
or analog inputs	max. 64	max. 128	max. 128	max. 128	max. 448
Digital outputs with process image	max 1024	–	–	–	max. 1024
Digital outputs without process image	max. 1024	max. 2048	max. 2048	max. 2048	max 7168
or analog outputs	max. 64	max. 128	max. 128	max. 128	max. 448

Flags	2048
S flags	32768
Timers	256
Counters	256
User memory capacity	640 or 1664 Kbyte RAM
Baud rate of the serial programmer port	9600 bps

Characteristic/Function	Value
Number of blocks Program blocks (PBs) Sequence blocks (SBs) Function blocks (FBs) Function blocks FXs) Data blocks (DBs) Data blocks (DXs) Organization blocks (OBs)	 256 256 256 256 256, 253 of which are programmable 256, 253 of which are programmable OB 1 to 39 (interfaces to the operating system)
Integral special-function organization blocks (OBs)	OB 121, 122, 124, 125, 131 to 133, 141 to 143, 150, 151, 153, 254, 255
Integral serial port	Programmer interface
Wiring backplane	S5 bus
Dimensions (W x H x D)	40.6 mm x 233.4 mm x 160 mm
Weight	Approx. 1 kg

Connector Pin Assignments

Please refer to Chapter 10 of the S5-135U/155U System Manual for a list of connector pin assignments on the CPU 948R's backplane connectors and front connector (programmer port). The pinout is the same as for the CPU 948.

I/O Operating Modes and Permissible I/O Modules

4

This chapter describes the possible I/O modes for the S5-155H (redundant, three-channel redundant, one-sided and switched) and lists the permissible modules in each case. The explanations concerning redundant I/Os in Sections 5.3 and 5.4 are of particular importance. You will also find the necessary wiring diagrams in that chapter. Standard function blocks FB 40/41 and 43 for analog value input/output are also discussed.

This chapter is vital for configuring and operating your I/O modules.

4.1 Overview

I/O Operating Modes

The S5-155H supports four different I/O operating modes:

- **Redundant I/Os**

  The module is present in **both** subunits under the **same** address. This mode offers a **high degree of fault tolerance.**

- **Three-channel redundant I/Os**

  There are three I/O modules. Two inputs are assigned to the same address and the third to either switched I/Os or to subunit A or B. However, the highest degree of fault tolerance is achieved if the third channel is assigned to switched I/Os.

- **One-sided I/Os**

  The module is assigned **exclusively** to one of the two subunits. If this subunit fails, the modules assigned to it also fail. This means that the fault tolerance of this configuration is **no higher** than that of an S5-155U.

- **Switched I/Os**

  The module can be operated by **either** of the two controllers. This mode offers **greater fault tolerance** than the S5-155U.

These modes can be combined in an S5-155H. Each I/O module can be configured individually.

Note

All four I/O modes – one-sided, switched, redundant and three-channel redundant – can be combined in one S5-155H.

I/O Types

When the digital/analog I/Os and CPs/IPs are configured with COM 155H, each process signal managed by the system program is assigned a specific type number.

The type number identifies

a) the signal type: digital, analog, input, output, CP, IP and

b) the operating mode: one-sided, switched, redundant, three-channel, redundant

The table below lists all configurable I/O types. Please also refer to the "COM 155H" description in this manual.

Type No.	Meaning		Fault tolerance
1	DI byte	one-sided	Standard (same as S5-155U)
2	DI byte	switched	Higher
3	DI byte	2-channel redundant	High
4	DI byte	3-channel redundant	Highest
8	DQ byte	one-sided	Standard
9	DQ byte	switched	Higher
10	DQ byte	redundant	High
11	DQ byte	redundant	High (with three readback DIs)
13	AI channel	one-sided	Standard
14	AI channel	switched	Higher
15	AI channel	2-channel redundant	High
16	AI channel	3-channel redundant	Highest
18	AQ channel	one-sided	Standard
19	AQ channel	switched	Higher
20	AQ channel	redundant	High
21	AQ channel	redundant with fault locating	Highest
24	CP/IP	one-sided	Standard
25	CP/IP	switched	Higher

4

Configuring the I/Os

Digital I/Os are configured **by byte**, analog I/Os **by word**. This means that you can assign each I/O byte/word the attribute "one-sided", "switched" or "redundant". When doing so, observe the following address restrictions:

I/O Type	Address Range	
One-sided digital I/Os	PY 0 ... 255	OY 0 ... 255
Switched digital I/Os	PY 0 ... 255	OY 0 ... 255
Redundant digital I/Os	PY 0 ... 127	
One-sided analog I/Os	PW 128 ... 254	OW 0 ... 254
Switched analog I/Os	PW 128 ... 254	OW 0 ... 254
Redundant analog I/Os	PW 128 ... 254	OW 0 ... 254

Switched digital and analog I/Os can also be used in other I/O areas (see Figure 4-1).

I/O Address Areas

Figure 4-1 provides an overview of the various I/O areas in the S5-155H for redundant, switched and one-sided I/Os.

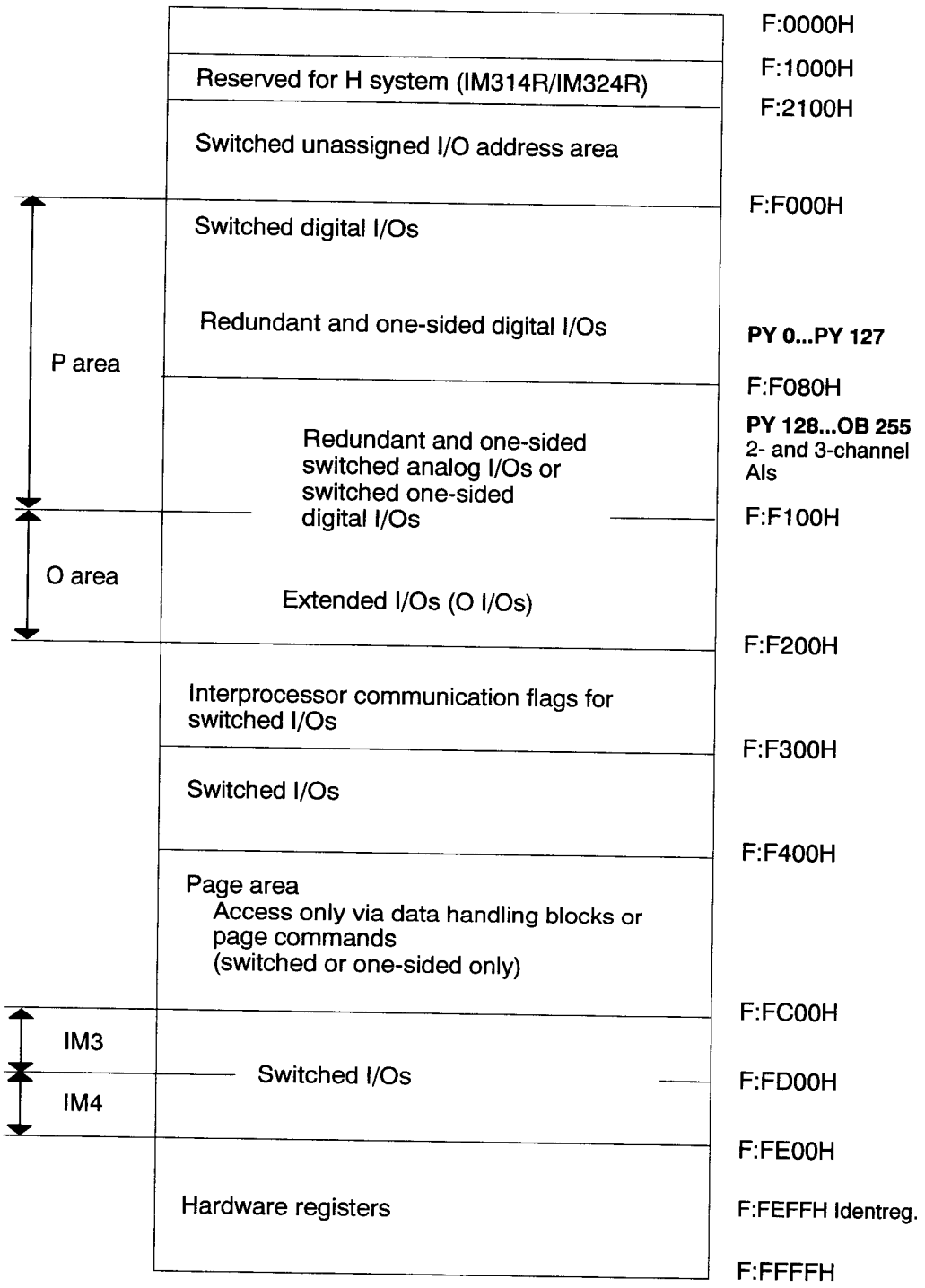


Figure 4-1 I/O Areas in the S5-155H

Updating the Process Input and Output Images

The process output image (redundant PIQ, switched PIQ, one-sided PIQ) is output after OB 1 has been processed. The S5-155H self-test then runs. This can last between 2 and 38 ms (test slice 2 ms * n), plus any time needed for interrupt servicing.

The process input image (redundant PII, switched PII, one-sided PII) is then read in. The PIIs of both subunits are exchanged and compared. OB 1 is then re-invoked (see Figure 2-2).

4.2 Redundant I/Os (Overview)

Redundant I/O (1-out-of-2)

In this mode, the input or output module has the **same** address in **both** subunits.

This mode offers the highest possible degree of fault tolerance, since failure of a central controller or an input or output module is tolerated (NON-STOP operation). The modules can be plugged into either the central controller or the expansion unit.

Figure 4-2 shows a redundant (two-channel) configuration.

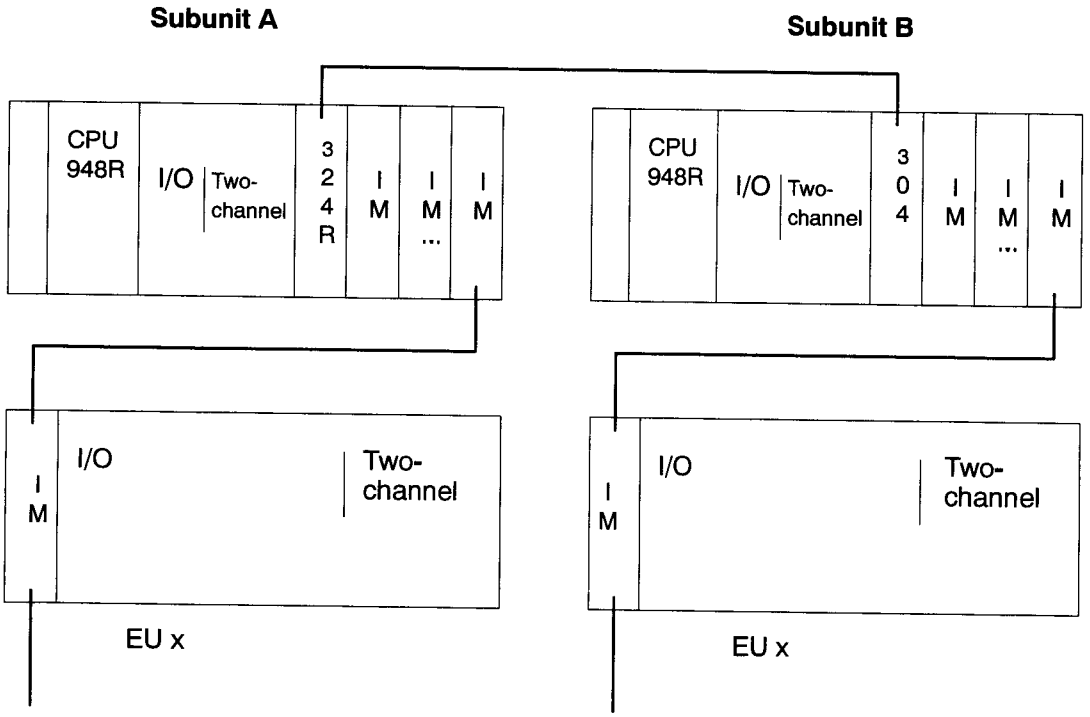


Figure 4-2 Redundant I/O Operation and Permissible Modules

Interface Modules and Expansion Units

In two-channel redundant I/O mode, the same interface modules and expansion units can be used as for the S5-155U (refer to the S5-135U/155U System Manual).

If the ET electronic terminator is being used in a redundant configuration, the following applies: If an ET 100 I/O byte must be passivated due to a timeout (QVZ), then the whole phase of this ET 100 is shut down.

Digital and Analog I/O Modules

"Redundant I/O" means the redundant input/output module in question is plugged into subunit A **and** subunit B, and both I/O modules have the **same I/O address and** have been configured with COM 155H as redundant.

All I/O modules which can be used in the S5-155U can also be used in the S5-155H.

I/Os

Redundant I/O modules are permitted in the following I/O address areas only (see Section 4.1):

- Redundant DIs/DQs: FF000 to FF07F (PY 0 to 127)
- Redundant AIs/AQs: FF080 to FF1FF (PW 128 to 254 and OW 0 to 254)

Please refer also to the configuring information for I/O modules in Section 4.1.

Note

If you want to operate specific redundant digital inputs or outputs as "NON-STOP DIs" or "NON-STOP DQs", please read the information presented below in "Locating facility (LF)".

Locating Facility (LF)

For every redundant digital input and every redundant digital output you want to operate as a NON-STOP DI or NON-STOP DQ, you must configure a special locating facility with which the 155H system program can locate errors quickly (see Figures 4-8/4-9).

"NON-STOP DI/DQ" means that the failure of the DI/DQ and its subsequent repair have no effect whatsoever on the process.

A locating facility (LF) for a NON-STOP DI or a NON-STOP DQ consists of

- a locating digital input (L-DI) and
- a locating digital output (L-DQ).

Overview of Redundant I/O Types

This section presents an overview in the form of keywords of the characteristics of the various redundant I/O types in the S5-155H.

- **Redundant DIs without error locating**
  - Error detection: By discrepancy monitoring and edge change monitoring
  - Error locating: None
  - Passivation: Passivation of the DI byte in the subunit in which a 'stuck at 0' error was found
- **Redundant DIs with error locating**
  - Error detection: By discrepancy monitoring
  - Error locating: By L-DQ
  - Passivation: Passivation of the defective DI byte
- **Redundant three-channel DIs**
  - Error locating: By the 2-out-of-3 method

- **Redundant DQs without error locating**

  In the case of 'stuck at 1' errors:

  - Error detection: By cyclic comparison of PIQs and readback DIs
  - Error locating: None
  - Passivation: Passivation of the readback DI; that is, testing of 0 to 1 edge no longer possible

  In the case of 'stuck at 0' errors:

  - Error detection:
  - Error location:

  } By testing a 0 to 1 edge change

  - Passivation: Passivation of the defective DQ byte

- **Redundant DQs with error locating**

  In the case of 'stuck at 1' errors:

  - Error detection: By cyclic comparison of PIQs and readback DIs
  - Error locating: By switching off the power supply to the group via L-DQ
  - Passivation: Passivation of the defective DQ byte and all other redundant DQs with the same group supply

  In the case of 'stuck at 0' errors:

  - Error detection:
  - Error locating:

  } By testing a 0 to 1 edge change or, at the latest, after approx. 10 hours when initial state = "1".

  - Passivation: None

    The defective DQ byte is reported only; access operations are not affected, and there is no further testing of the DQ byte.

- **Redundant AIs**
  - Error detection: By analog value discrepancy monitoring; minimum or maximum value may be selected on error.

  - Error locating:
  - Passivation:

  } Error-dependent; see Section 4.4

- **Redundant three-channel AIs**

- Error detection:
- Error locating:
- Passivation:

} By analog value discrepancy monitoring

- **Redundant AQs**

- Error detection:
- Error locating:
- Passivation:

} By reading back the analog output value

Note:
More detailed information on all of the I/O types listed above, and in the same order, can be found in Sections 4.3 and 4.4.

Detecting Errors on Digital Output Modules

The table below tells you the amount of time needed to detect an error on digital output types 10 and 11.

	Error	Type 10 with Readback DI	Type 11 with 3 Readback DIs
1.	'Stuck at 1' on DQ module. When PIQ intermittent.	After two 0 → 1 edge changes	After two 0 → 1 edge changes
2.	'Stuck at 1' on DQ module. When PIQ = 0.	After TI at latest	After TI at latest
3.	'Stuck at 1' on DQ module. When process image = 1.	Through inspection	Immediately preceding the L-DQ test = every 10 hours
4.	'Stuck at 0' on DQ module or wirebreak in DQ module connection. When PIQ intermittent.	After two 0 → 1 edge changes or *) in L-DQ test = every 10 hours (fault has short-term effect[1)] on process)	After two 0 → 1 edge changes or *) immediately preceding the L-DQ test = every 10 hours
5.	'Stuck at 0' on DQ module or wirebreak in DQ module connection. When PIQ = 0.	Through inspection	Through inspection
6.	'Stuck at 0' on DQ module or wirebreak in DQ module connection. When PIQ = 1.	In the L-DQ test (= every 10 hours) (fault has short-term [1)] effect on process)	Immediately preceding the L-DQ test = every 10 hours
7.	'Stuck at 1' after decoupling diode (fault cannot be automatically rectified). When process image = 0.	After TI at latest	After TI at latest
8.	'Stuck at 1' after decoupling diode (fault cannot be automatically rectified). When process image = 1.	Through inspection	Through inspection
9.	Wirebreak in or after the decoupling diode. When PIQ intermittent.	After two 0 → 1 edge changes or *) immediately after the L-DQ test = every 10 hours (fault has short-term[1)] effect on the process)	After two 0 → 1 edge changes or *) immediately following L-DQ test = every 10 hours (fault has short-term[1)] effect on the process)
10.	Wirebreak in or after decoupling diode. When process image = 0.	Through inspection	Through inspection
11.	Wirebreak in or after decoupling diode (the wires must be redundant up to the actuator and must be run from the actuator to the readback DI). When process image = 1.	In the L-DQ test = every 10 hours (fault has short-term [1)] effect on the process)	In the L-DQ test = every 10 hours (fault has short-term [1)] effect on the process)
12.	Short-circuit to 0 after decoupling diode (fault cannot be automatically reported, but can be located). When PIQ = 1.	After TI at latest	After TI at latest

TI = 2 * T_R or 2 * PLC scan time (whichever is higher)

*) Depending on which event occurs first

1) Short-term effect on the process ('short-term' means 1 * t_R + 2 * basic clock rate for timed interrupts, without regard to the PLC scan time)

T_R = Configured readback delay time t_R = Actual readback delay time

4.3 Redundant Digital Inputs/Outputs (DIs/DQs)

Redundant DIs without Error Locating Facility

Two-channel (1-of-2) DIs without error locating facility detect faults, but do not locate them. This means that these DIs **cannot** be operated as "NON-STOP" DIs.

There is a redundant digital input in both subunit A and subunit B. The system program compares these DIs cyclically to make sure that their signal states are identical; this comparison is made during updating of the PII. If the system program discovers redundant DIs with different signal states, these DIs are flagged and the configured discrepancy timer started. As long as the timer is running, the last identical value is maintained as signal state.

If the signal states of the DIs are still different after the discrepancy time has expired, an appropriate entry is made in the error DB, and the system program waits for the next edge change. Until this edge change takes place, the last identical value continues to be retained as signal state. After the edge change, the signal state of the DI on which the edge change took place becomes the final valid signal state. The other side is passivated.

The signal states of the redundant DIs are also compared in cases of direct I/O access. In the event of a discrepancy, the last identical value is taken as signal state.

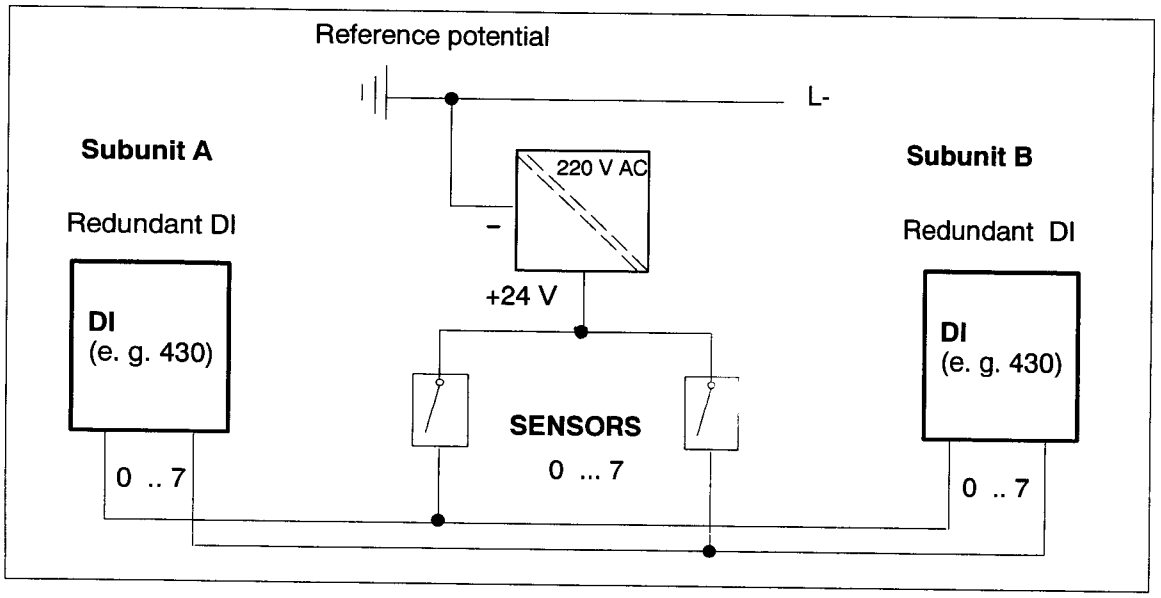


Figure 4-3 Redundant Two-Channel DIs without Error Locating Facility

Redundant DIs with Error Locating Facility

The 1-out-of-2 DI with LF both detects and locates faults. With the aid of the circuitry shown in Figure 4-4, the system locates the defective DI module in the event of a fault. The following sequence is then put into effect when the discrepancy time has elapsed:

- A "0" signal is output to both L-DQs (subunits A and B), thus switching off the sensor power supply.
- A "0" must be read back from both DIs after the specified readback delay time has elapsed. A "1" at a DI indicates a defect in the module.
- A "1" signal is output to both L-DQs.

The defective module is reported and the DI byte passivated; that is, this subunit's DI byte will no longer be accessed (one-sided operation). Fault location can take several PLC cycles, depending on the configured readback delay time. During this time, the last valid process I/O image is transferred in the case of direct access to the relevant DI bytes. "Relevant" DI bytes are all DI bytes sharing the same group power supply.

The term "group" designates all sensors for redundant DIs or DQs supplied by the same L-DQ. The smallest possible group consists of one redundant byte, the largest possible group of all redundant DIs in an S5-155H.

Testing the Error Locating Facility

The L-DIs of the two-channel 1-out-of-2 are tested for 'stuck at 0' and 'QVZ' (timeout) once in each test cycle (approx. every 5 minutes).

The L-DIs and L-DQs of the two-channel 1-out-of-2 DIs are tested for 'stuck at 0' every 10 hours by setting the L-DQs to 0 page by page. They are tested for 'stuck at 1' only in the master's Restart routine by setting the L-DQs to 0 in both subunits. Also refer back to the table entitled "Detecting Errors on Digital Output Modules".

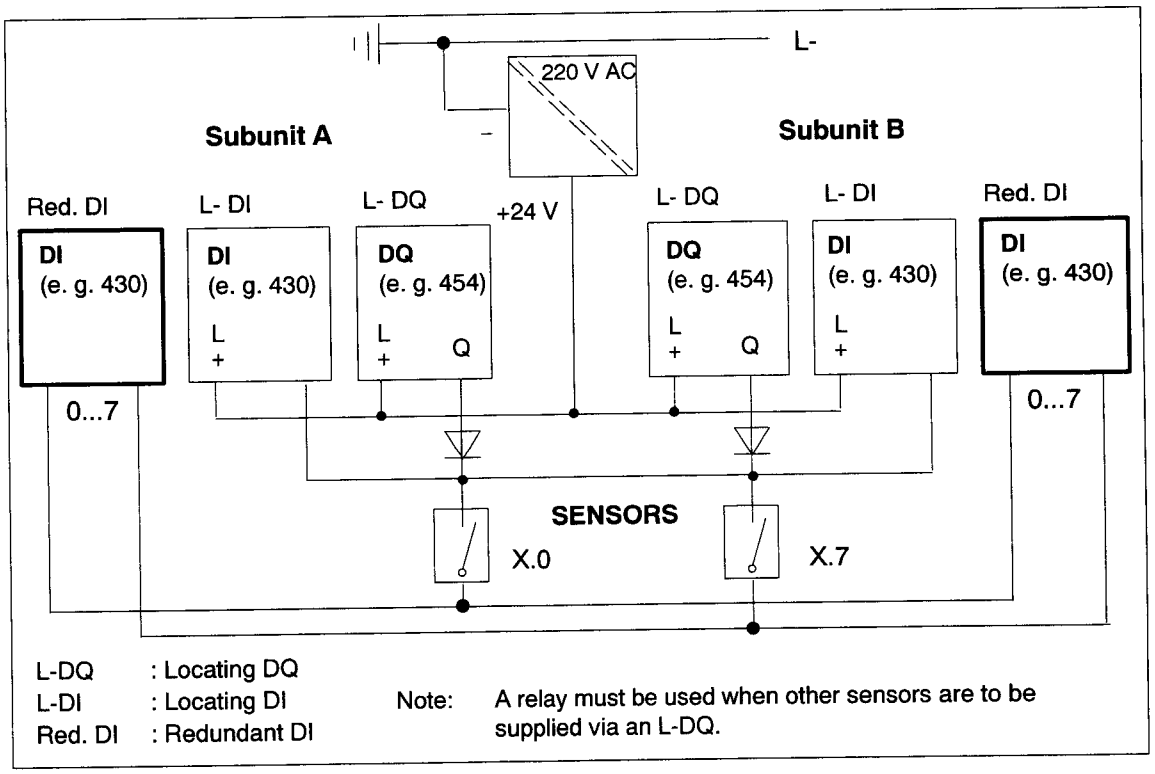


Figure 4-4 Redundant Two-Channel DI with Error Locating

Redundant Three-Channel DIs

Three-channel (1-out-of-3) DIs detect and locate faults. This means that these DIs can be operated as "NON-STOP DIs".

Each subunit is equipped with one redundant digital input, and the third is assigned to switched I/Os or to subunit A or B. Assigning the third DI to the switched I/Os provides a higher degree of fault tolerance than assigning it to subunit A or B.

The 155H system program compares the three DIs cyclically to make sure that their signal states are identical. This comparison takes place during updating of the process input image. If the system program discovers DIs with different signal states, these DIs are flagged and the specified discrepancy timer started. As long as this timer is running, the relevant standard value is taken as signal state, depending on whether one or three sensors were configured. If only one sensor was configured, the standard value is based on a 2-out-of-3 decision. If three sensors were configured, the last identical value is retained as signal state.

If the signal states of the DIs are still different after the discrepancy time has elapsed, the fault is reported. The result of the 2-out-of-3 decision is given as valid signal state. The byte is passivated. The DI continues on a 1-out-of-2 basis until the fault has been eliminated and the byte depassivated.

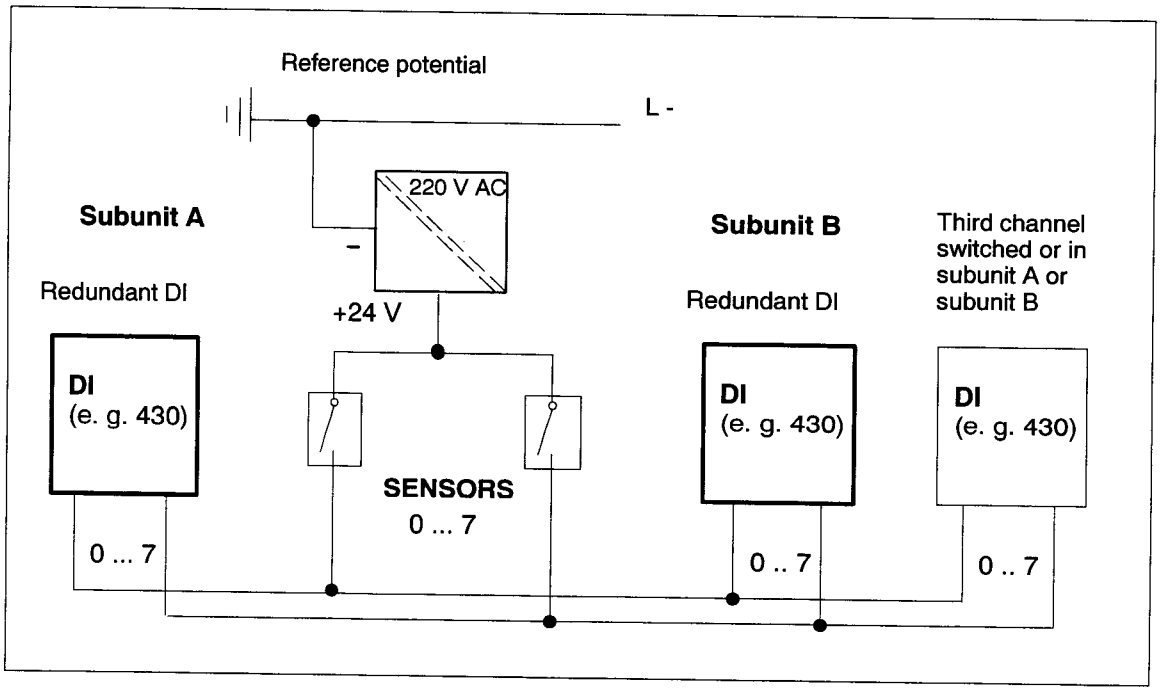


Figure 4-5 Redundant Three-Channel DI with One Sensor

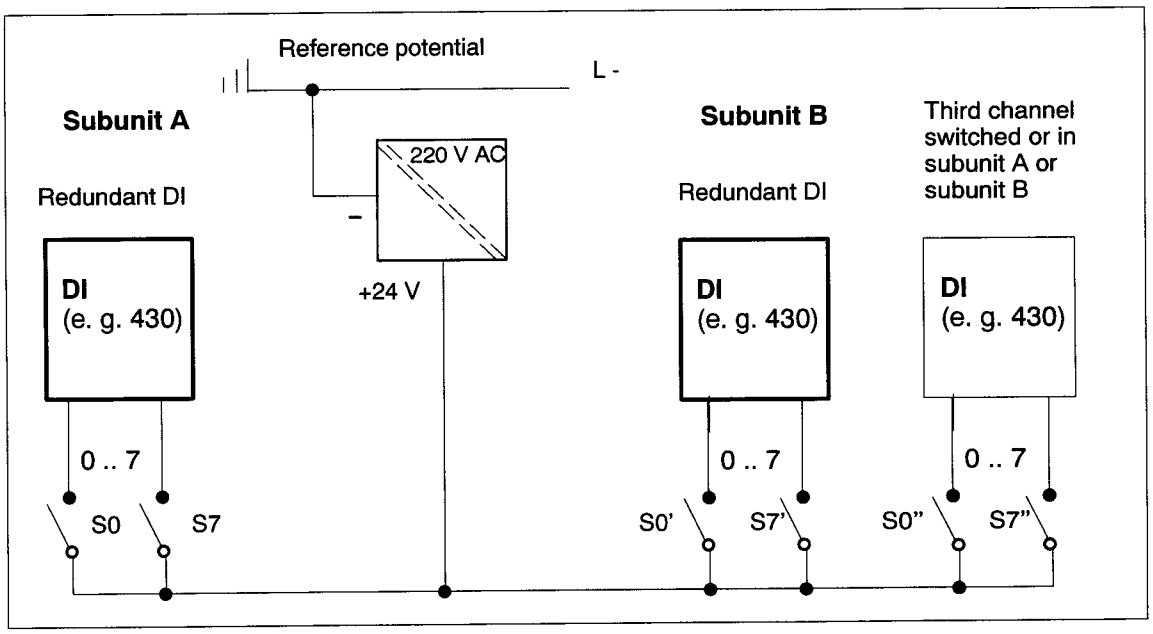


Figure 4-6 Redundant Three-Channel DI with Three Sensors

Direct I/O access to a three-channel DI is permitted.

The standardized value of the three digital inputs is supplied as the result.

The use of three sensors provides the highest possible degree of fault tolerance, as sensor faults can also be detected, located and passivated.

The maximum amount of time needed to locate a fault is:

2 x configured discrepancy time +

2 x PLC scan time.

Configuring Redundant Process Interrupts (DI 0)

1. Configure DI 0 as a redundant digital input without error locating (and wire accordingly).

   The system program uses the time configured for DI 0.0 as the discrepancy time for all DI 0 bits. Times specified for DI 0.1 to DI 0.7 are irrelevant.

   The maximum discrepancy time configurable for DI 0 is 1.0 s.

1. Enter "Interrupt DI": YES in COM 155H's operating system form.
2. Specify "Process interrupts:"YES and "Timed interrupt servicing":YES in data block DX 0. Specify 10 ms as "Basic clock rate for timed interrupt servicing".

Functions, Principle of Operation and Error Handling

Only the master subunit has access to DI 0. The standby subunit checks itself and the master every 10 ms for a 'stuck at 0' or 'stuck at 1' error.

If a 'stuck at 0' or 'stuck at 1' error triggers an edge on the DI, the interrupt OB is invoked as a result. An additional DI byte wired to DI 0 will allow the user to detect the 'wrong' edge change.

The 'correct' edge on a properly functioning DI always results in invocation of an interrupt OB. The system program thus ensures that no interrupts will be lost due to 'stuck at 0/1' errors.

If a fault occurs in the master subunit, the system program initiates a standby-master transfer as soon as the discrepancy time has elapsed. The fault is reported, the defective DI 0 passivated, and the 'AGF' bit (PLC fault) set in the H flag word (see Section 8.5).

If a fault occurs in the standby subunit, the fault is reported, the defective DI 0 passivated and the 'AGF' bit set in the H flag word as soon as the discrepancy time has elapsed.

Wiring DI 0

Wire DI 0 as a redundant DI without error locating. Even without error locating, 'stuck at 0' and 'stuck at 1' errors will be located (see above). Unused inputs must be connected to ground.

Redundant DQs without Error Locating Facility (LF)

Two-channel (1-out-of-2) DQs without error locating facility can only detect errors/faults, but cannot locate them, or only to a limited degree. The system treats this type of DQ as follows:

- Reading back of the digital output values, taking into consideration the configured readback delay time. This makes it possible to detect, but not locate, 'stuck at 1' errors.

'Stuck at 0' errors are detected only after the next edge change from 0 to 1:

- First, a "1" signal is output in one subunit (subunit A, for instance) while a "0" signal continues to be output in subunit B.
- After the configured readback delay has elapsed, the readback must show a "1" signal. If it does not, the error is located and reported.
- This DQ test is executed alternately in the other subunit on every edge change from 0 to 1.

Note:
When configuring with COM 155H, you must enter the DQ readback delay time, since the different digital output modules have different signal propagation times.

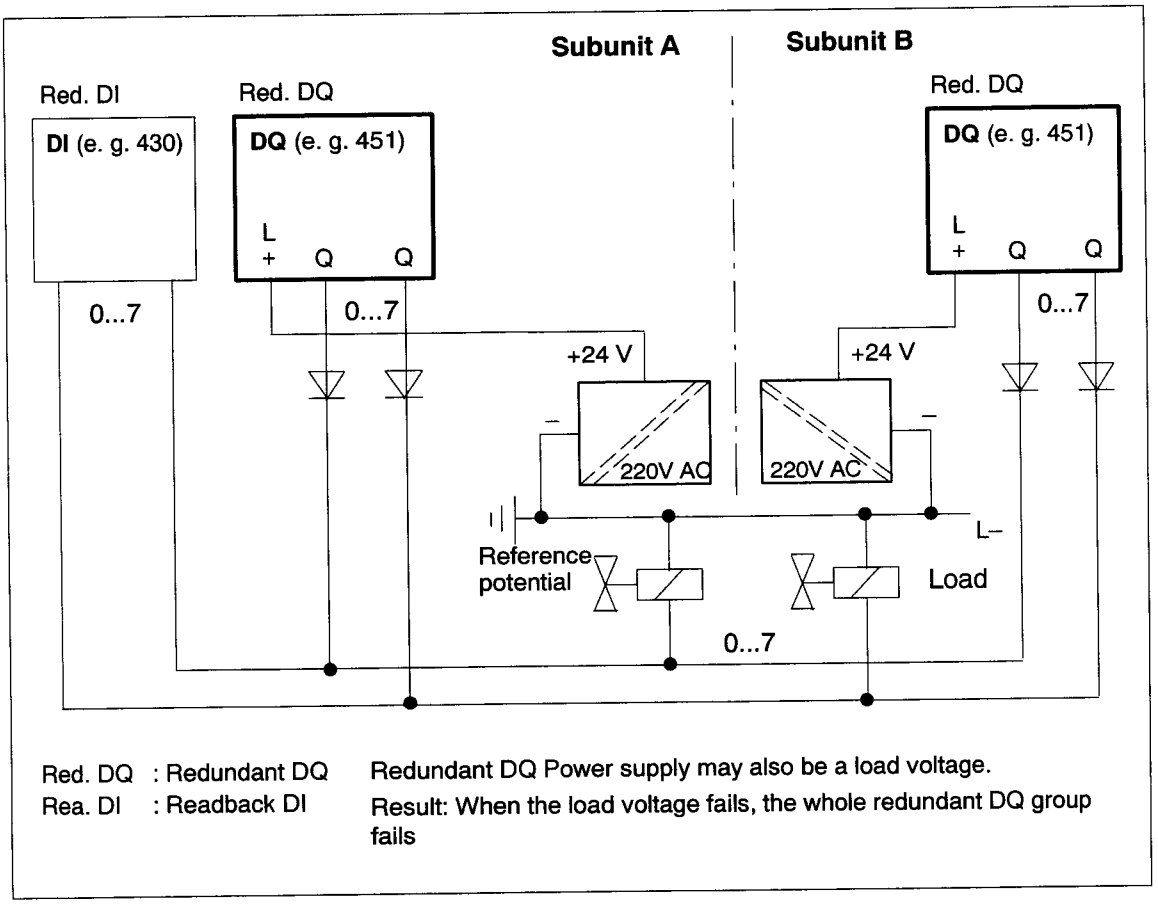


Figure 4-7 Two-Channel Redundant DQ without Error Locating

Redundant DQs with Error Locating Facility (DQ Type 10)

Two-channel DQs (1-out-of-2) for intermittent (frequently switched) outputs can both detect and locate faults/errors. In the event of a 'stuck at 1' signal, a DQ of this type is passivated/isolated by the L-DQ in its controller (subunit), which outputs "0" (shut down). In the case of a 'stuck at 0' signal, the error is reported only. This means that this digital output can be operated as NON-STOP DQ (Figure 4-8).

The system program treats this type of DQ as follows:

- Reading back of the digital output values, taking into account the configured readback delay time. This makes it possible to detect 'stuck at 1' errors and to localize them by shutting down the group power supply.

'Stuck at 0' errors cannot be detected until the after the next edge change from 0 to 1:

- First, a "1" signal is output in one of the subunits (subunit A, for instance) while subunit B continues to output a "0" signal.
- The readback value must be "1" as soon as the readback delay time has elapsed. Otherwise, the error is located and reported.
- This DQ test is executed alternately in the other subunit on each edge change from 0 to 1.

155H System Program Response

No further DQ tests are performed on the relevant DQ byte in the event of a 'stuck at 0' error, nor is the L-DQ tested for 'stuck at 1'. In order to enhance the fault tolerance, the byte is not passivated, and can continue to be accessed.

In the case of 'stuck at 1' errors, the bad DQ byte and the associated readback DI are passivated. The relevant DQ byte is no longer accessed. The group power supply is shut down via the L-DQ, which passivates all redundant DQs connected to this group supply (one-sided operation).

A 'stuck at 0' error for an initial status of "1" (during execution of the "L-DQ for DQ test") is detected a maximum of T_{test} times following the first possible readback of the 'stuck at 0' error and the L-DQ test aborted.

T_{test} is a maximum of 3 * the basic clock rate for timed interrupts. In order to ensure that an error of this type affects the actuator for as brief a time as possible, the basic clock rate for timed interrupts in DX 0 should be set to 1 * 10 ms.

When type 10 DQs are used, an error of this kind affects the actuator for a maximum period of T_{test}.

Testing the Error Locating Facility

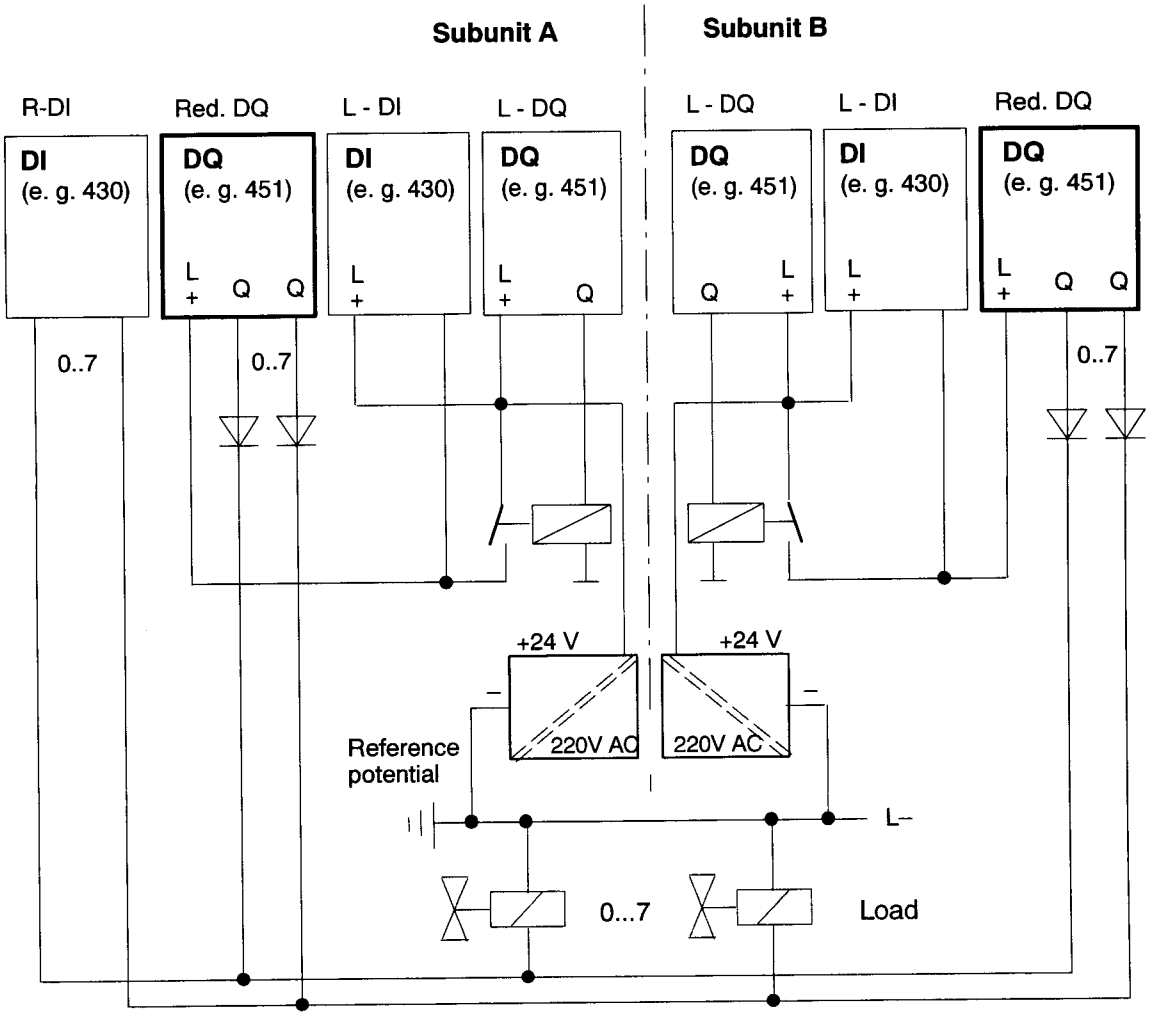
The L-DIs and L-DQs of the two-channel 1-out-of-2 DIs are tested once per test cycle (approx. every 5 minutes) for 'stuck at 0' and 'QVZ' (timeout).

The L-DQs of the two-channel 1-out-of-2 DQs may and must be set to zero page by page only. 'Stuck at 1' errors are detected every 10 hours by setting one L-DQ at a time to zero on a page-by-page basis.

Type 10 DQs with L-DQs are designed in such a way that a 'stuck at 0' error detected during an L-DQ test affects the process for only a very brief span of time.

If a 'stuck at 0' error is present (during an "L-DQ for DQ test") on a two-channel 1-out-of-2 DQ with an initial state of "1", it is detected no more than two basic timed interrupt clock pulses after the first possible readback of the 'stuck at 0' error and the L-DQ test aborted. The L-DQ is then immediately set to "1" so that the non-errored DQ will output "1". The 'stuck at 0' error is reported.

Please also refer to the table entitled "Detecting Errors on Digital Output Modules".



L-DQ : Locating DQ
L-DI : Locating DI
Red. DQ : Redundant DQ
R-DI : Readback DI

A load power supply may also be used.
Result: If the load supply fails, the entire redundant DQ group fails

Figure 4-8 Two-Channel Redundant DQs with Error Locating Facility (Type 10)

Redundant DQ with Error Locating Facility and 3 Readback DIs (DQ Type 11)

The two-channel (1-out-of-2) DQ for non-intermittent (infrequently switched) outputs detects and locates errors. The system program handles this type of DQ as follows:

- Readback of the digital output values, taking into account the configured readback delay time. This makes it possible to detect 'stuck at 1' errors, and then to locate them by shutting down the group power supply.

'Stuck at 0' errors are not detected until after the next edge change from 0 to 1:

- First, a "1" signal is output in one subunit, (subunit A, for instance) while a "0" signal continues to be output in the other subunit (subunit B in this case).
- The readback value must be "1" once the configured readback delay time has elapsed. Otherwise, the error is located and reported.
- This DQ test is executed alternately in the other subunit on each edge change from 0 to 1.
- In addition, with the aid of the R-DIs in subunits A and B, the DQ is tested for 'stuck at 0' errors every 10 hours and 5 minutes after each passivation. This makes it possible to detect a 'stuck at 0' error in the DQ module even without an edge change.

I/O Type 11

A locating facility must be configured for type 11 (Figure 4-9). This circuit is required to allow NON-STOP operation of a non-intermittent DQ.

If a 'stuck at 0' error is present (during execution of the "L-DQ for DQ test") on a DQ module with an initial state of "1", it is detected prior to execution of the L-DQ test and the L-DQ test is suppressed. The 'stuck at 0' error is reported.

If, in the case of initial state "1", a 'stuck at 0' error is present in or after the decoupling diode (during execution of the "L-DQ for DQ test"), it is detected no more than T_{test} after the first possible readback of the 'stuck at 0' error and the L-DQ test aborted.

T_{test} is a maximum of 3 * the basic clock rate for timed interrupts. In order that an error of this kind affect the actuator as briefly as possible, the basic clock rate for timed interrupt should be set to 1 * 10 ms in DX 0.

155H System Program Response

No further DQ tests are performed on the relevant DQ byte in the case of a 'stuck at 0' error. To increase fault tolerance, the byte is not passivated and will continue to be accessed.

The bad DQ byte is passivated in the case of 'stuck at 1' errors. The byte is no longer accessed. The group power supply is shut down via the L-DQ, thus passivating all redundant DQs connected to this group supply (one-sided operation).

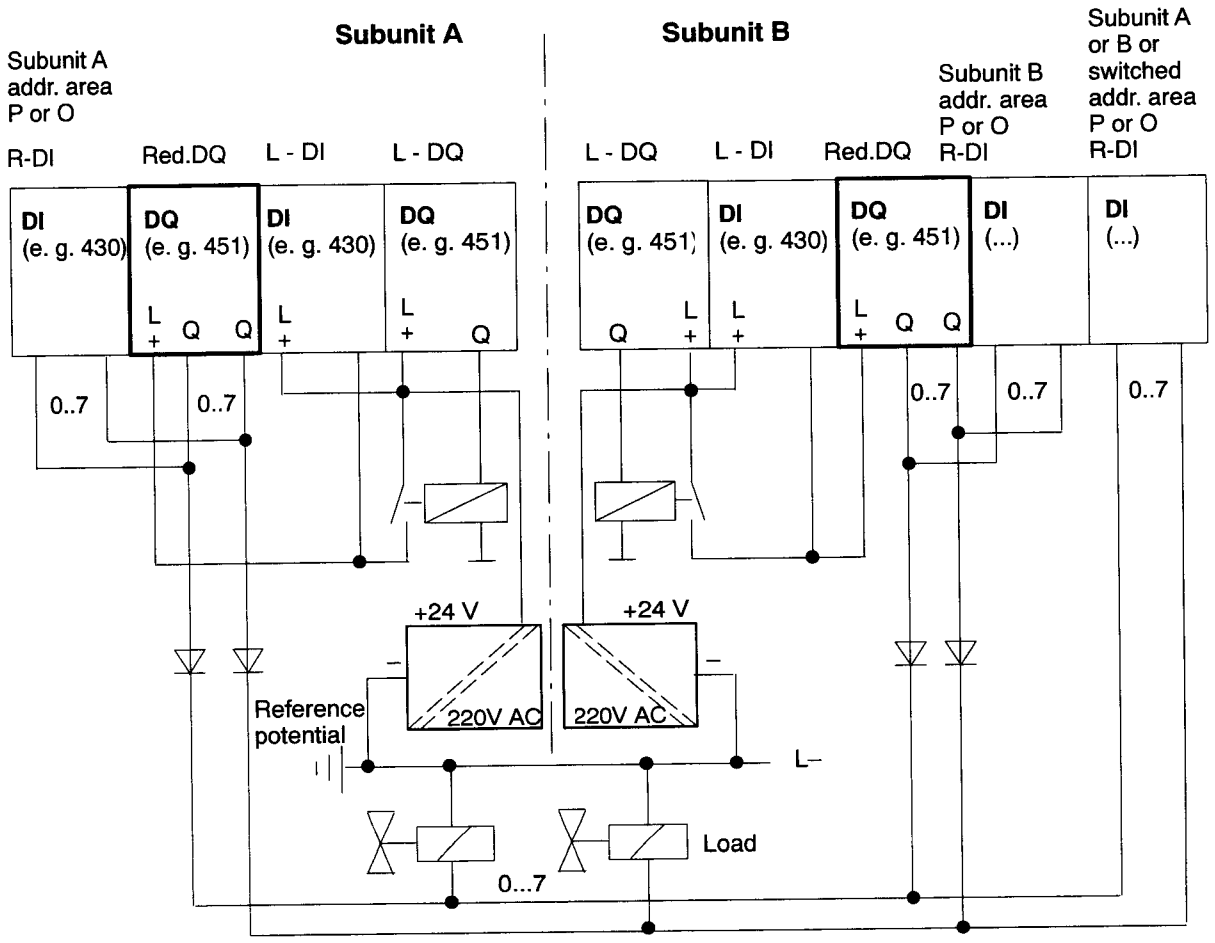
Testing the Error Locating Facility

The L-DIs and L-DQs of two-channel 1-out-of-2 DIs are tested once per test cycle (approx. every 5 minutes) for 'stuck at 0' and 'QVZ' (timeout).

The L-DQs of two-channel 1-out-of-2 DQs may and must be set to zero page by page only. 'Stuck at 1' errors are detected every 10 hours by setting one L-DQ at a time to zero page by page.

If, when the initial state is "1", a 'stuck at 0' error is present **in the DQ module** (during the "L-DQ for DQ" test), it is detected **prior to** execution of the L-DQ test and the test suppressed. The 'stuck at 0' error is reported.

If, in the case of an initial state of "1", a 'stuck at 0' error is present **in or after the decoupling diode** (during execution of the "L-DQ for DQ test"), the error is detected no more than two timed interrupt basic clock pulses after the first possible readback of the 'stuck at 0' error and the L-DQ test aborted. The L-DQ being tested is then immediately set to "1" in order to reduce the duration of the effect on the process to a minimum. The 'stuck at 0' error is reported. (Also refer to the table entitled "Detecting Errors on Digital Output Modules").



L-DQ : Locating DQ
L-DI : Locating DI
Red. DQ : Redundant DQ
R-DI : Readback DI

A load voltage supply may also be used.
Result: If the load voltage fails, the entire redundant DQ group fails

Figure 4-9 Redundant Non-Intermittent DQ with Error Locating Facility (Type 11)

4.4 Redundant Analog Input/Outputs (AIs/AQs)

Direct I/O Access

The "L PY" operation for redundant analog inputs is **not** permitted, and results in a Transfer error (TLAF).

I/O access operations L PW and L OW to three-channel redundant analog inputs are permitted. These operations supply the standardized value. If only two channels are still active at the time, these operations supply the higher of the two analog input values.

Redundant AIs: FB 40

The two-channel (1-out-of-2) analog inputs detect errors, but do not always locate them.

Function block FB 40 "H-RLG:AE" is provided expressly for the purpose of reading in analog values. It is part of the COM 155H package, and is on the diskette in program file S5CR70ST.S5D. All other required parameters, such as discrepancy value and discrepancy time, must be configured with COM 155H.

FB 40 can be used for modules 460 and 465. It reads in the first two analog values and replaces the raw values. If a module reports a wirebreak, overflow or range violation, that module is passivated and the error reported. From that point on, operation is one-sided, using only the other module. The simultaneous occurrence of an overflow and a range violation does not result in passivation.

The input and output parameters for the function block are identical to those of FB 40 "RLG:AE" for the S5-155U. In contrast, the test for range violations in modules which are configured as redundant takes into account the specified upper and lower range limits.

Execution Time for "Function Blocks"

Depending on the operations used, the execution times for "H" function blocks may be higher than those for standard function blocks:

FB 40 H-RLG:AE, for two-channel redundant AIs, by approx.:	750 µs

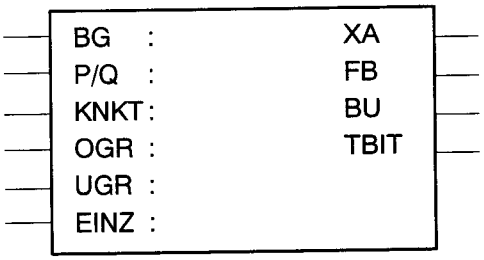
Calling the FB for 2-Channel Redundant AIs (FB 40)

Function block FB 40 reads analog value XE from two analog value inputs and, based on its nominal range, provides a proportional output value XA within the specifiable range limits UGR (lower limit) and OGR (upper limit). The analog value can be read in either by the cyclic or selective sampling method. The FB number must not be changed during loading.

STEP 5 program

```
      :   JU FB 40
NAME  :   H-RLG:AE
BG    :
P/Q   :
KNKT  :
OGR   :
UGR   :
EINZ  :
XA    :
FB    :
BU    :
TBIT  :
```

Graphic representation
FB 40



4

Description of input and output parameters

Name	Parameter Type	Data Type	Description	Remarks
BG	D	KF	Module address	P/Q = P: BG = 128 to 240 P/Q = Q: BG = 0 to 240
P/Q	D	KS	I/O area	P/Q = P: P area P/Q= Q: O area
KNKT	D	KY	Channel number KN Channel type KT	KN = 0 to 15 KT = 3 to 6 (see Note)
OGR	D	KG	Upper limit of the output value	–1701412+39 $\leq$ +1701412+39
UGR	D	KG	Lower limit of the output value	–1701412+39 $\leq$ +1701412+39
EINZ	I	BI	Selective sampling	EINZ = 0: Cyclic sampling
XA	Q	D	Address for output value XA	Scaled value between UGR and OGR (see Note)
FB	Q	BI	Bit address for "Wirebreak" bit	0 = No wirebreak 1 = Wirebreak
BU	Q	BI	Bit address for "Range violation" bit	0 = No range violation 1 = Range violated
TBIT	Q	BI	Bit address for "Analog input active, selective sampling" bit	0 = Not active 1 = Active

Parameter Type	Data Type	Permissible Actual Operands
I, Q	BI for an operand with bit address	I, Q, F
	D for an operand with doubleword address	ID, QD, FD, DD

Notes on Channel Type KT

The permissible analog input modules can provide the analog value in four different notations. You must choose the one you want by setting the KT parameter accordingly.

KT = 3 : Absolute value between 4 and 20 mA
KT = 4 : Unipolar representation
KT = 5 : Bipolar absolute value
KT = 6 : Bipolar fixed-point number

If the KT parameter is set to a value less than 3, the function block uses KT = 4; if it is set to a value exceeding 6, the function block uses KT = 6. The KT parameter must be in agreement with the method of representation set on the module itself.

Notes on Output Value XA

The input value (XE) read in from the analog input module is converted according to the formulas below depending on channel type parameter KT.

OGR : Upper limit value UGR : Lower limit value
XE : Input value XA : Output value

$$KT = 3 \quad XA = \frac{UGR \times (2560 - XE) + OGR \times (XE - 512)}{2048}$$

$$KT = 4 \quad XA = \frac{UGR \times (2048 - XE) + OGR \times XE}{2048}$$

$$KT = 5/6 \quad XA = \frac{UGR \times (2048 - XE) + OGR \times (XE + 2048)}{4096}$$

Notes on UGR/OGR:

Range limits for the output value

The analog value can be represented as a physical value if suitable range limits are selected.

Example:

Analog Value Range	Phys. Value	Range Limits	Resolution
0 to 10 V	2 to 150 °C	2000000+01 1500000+03	0.1 °C

Notes on Range Violation (BU) Bits

Range violation BU

In the case of modules which were configured as one-channel modules, a violation of the nominal analog value range is reported in BU. If the analog value is within the overflow range (analog value > +4096 or < –4096 units), it is limited to +4096 or –4096.

In the case of redundant modules, the upper and lower limiting values configured with COM 155H are used to check for range violations. If the analog value lies within the overflow range (analog value > +4096 or < –4096), the "Wirebreak" and "Range violation" bits are set, and the analog value limited to + or – 4096 units.

In the case of channel type 3 (4 to 20 mA), the BU bit is also set for values from 3 to 4 mA.

Wirebreak

In the event of a wirebreak, XA is set to 0.

Notes on Selective Sampling

For selective sampling, the function block initiates encoding and sets activity bit TBIT to 1 for its duration. Only one channel on a module may be sampled at any given time. Calls for the relevant function blocks should therefore be interlocked with the aid of the activity bit.

All AI addresses must also be configured as AQs of the same type.

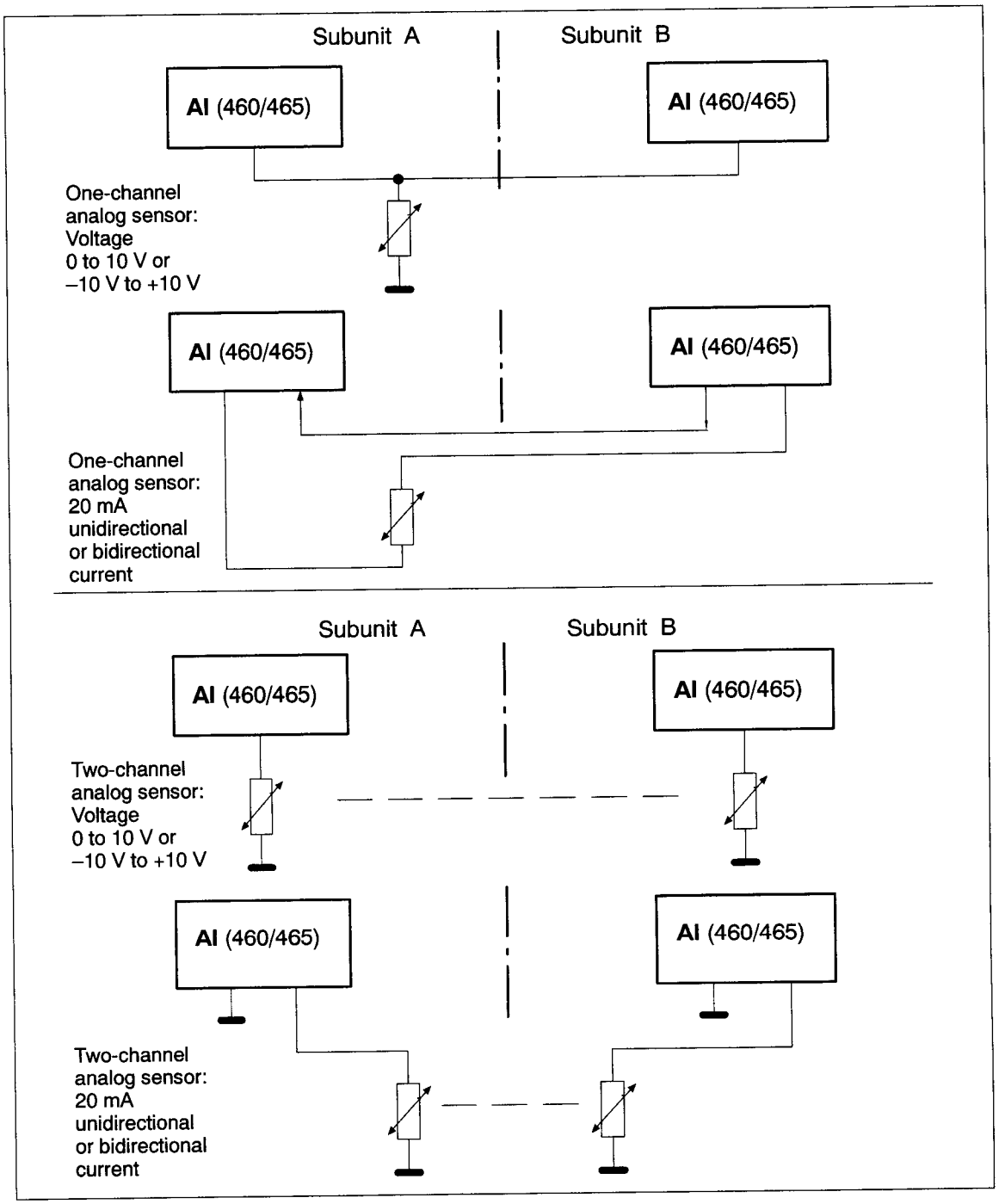


Figure 4-10 Redundant 1-out-of-2 AI, One/Two-Channel with One or Two Sensors

Notes on Error Detection

Both analog values are checked for discrepancies. The permissible discrepancy value is calculated by adding a relative value (a percentage of the maximum value of the two analog values) to an absolute value (in units corresponding to the analog value's notation):

$$D_{perm.} = ABS + \frac{REL \times RAWV(max)}{100}$$

$D_{perm.}$:	Permissible analog value discrepancy
ABS:	Absolute component of the configured discrepancy
REL:	Relative component of the configured discrepancy
RAWV(max):	The higher of the two instantaneous analog values

Example

The following was configured with COM 155H:

ABS:	100
REL:	10%
Preferred value:	Max.
Actual analog value subunit	A: 1000
Actual analog value subunit	B: 980
Actual discrepancy:	D = 1000 – 980 = 20

Permissible discrepancy $D_{perm.} = 100 + \frac{10 \times 1000}{100} = 200$

It follows that the actual discrepancy lies within the permissible range. For additional calculations, the value 1000 will be used for XE.

When FB 40 is called, the analog values are read in from the two subunits, exchanged, and standardized. If a one-sided, locatable error (timeout, wirebreak) occurs, the module is passivated; that is, it will no longer be accessed (one-sided operation).

If the 155H system program detects a discrepancy error, it checks one of the subunits for an overflow or range-violated condition. If it finds one, the module is passivated. Otherwise, the standby controller's module is passivated when the specified discrepancy time has elapsed.

If the two modules report different errors, passivation is prioritized as follows:

1. Timeout (QVZ)
2. Wirebreak
3. Overflow
4. Range violation.

If FB 40:H-RLG AE detected only a range violation or overflow condition but not a discrepancy, the configured standardized analog value is passed, together with the "Range violation" (BU) or "Overflow" (BU) bit. If FB 40:H-RLG AE detects a wirebreak in both modules, it sets the "Wirebreak" bit (FB).

Direct I/O access to redundant analog inputs via STEP 5's "L PW" operation returns a standardized value as the result. Depending on what was specified during configuring (min/max), this value is either the lower or higher **raw value**.

The "L PY" operation may **not** be used on redundant analog inputs, and results in a Transfer error (TLAF).

Three-Channel Redundant AIs: FB 43

Three-channel (3-out-of-1) AIs always detect and locate errors.

Standard function block FB 43 "3-RLG:AE" is provided for reading in three-channel analog values. It is part of the COM 155H package, and is on the diskette in program file S5CR70ST.S5D.

All other required parameters, such as the discrepancy value, the discrepancy time and the address of the third channel, must be configured using COM 155H.

The function block can be used for modules 460 and 465. Its input and output parameters are identical to those of standard FB 40 "RLG:AE" for the S5-155U. The only difference is that the specified upper and lower limit values are used to check for range violations on those modules configured as redundant modules.

Execution Time for "Function Blocks"

Depending on the operations used, the execution time of the "H" function blocks may be longer than that of the standard function blocks:

FB43: 3-RLG:AE, for three-channel redundant AIs, by approx.: 800 µs

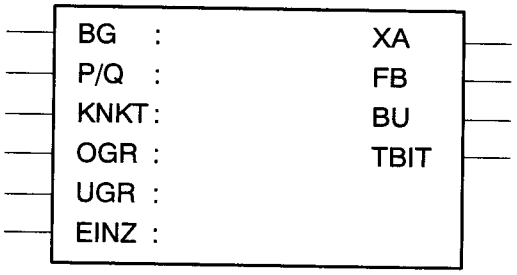
Calling the FB for 3-Channel Redundant AIs

Function block FB 43 reads three analog values XE from three analog value inputs and provides a proportional output value XA, based on its nominal range, between UGR (lower limit) and OGR (upper limit). The analog value can be read in cyclically or selectively. The FB number may be changed during loading.

STEP 5 program

```
      :   JU FB 43
NAME  :   3-RLG:AE
BG    :
P/Q   :
KNKT  :
OGR   :
UGR   :
EINZ  :
XA    :
FB    :
BU    :
TBIT  :
```

Graphic representation
FB 43



Description of Input and Output Parameters

Name	Parameter Type	Data Type	Description	Remarks
BG	D	KF	Module address	P/Q = P: BG = 128 to 240 P/Q = Q: BG = 0 to 240
P/Q	D	KS	I/O area	P/Q = P: P area P/Q = Q: O area
KNKT	D	KY	Channel number KN, Channel type KT	KN = 0 to 15 KT = 3 to 6 (see Note)
OGR	D	KG	Upper limit of the output value	$-1701412+39 \leq +1701412+39$
UGR	D	KG	Lower limit of the output value	$-1701412+39 \leq +1701412+39$
EINZ	I	BI	Selective sampling	EINZ = 0: Cyclic sampling
XA	Q	D	Address for output value XA	Scaled value between UGR and OGR (see Note)
FB	Q	BI	Bit address for "Wirebreak" bit	0 = No wirebreak 1 = Wirebreak
BU	Q	BI	Bit address for "Range violation" bit	0 = No range violation 1 = Range violated
TBIT	Q	BI	Bit address for "Analog input active, selective sampling" bit	0 = Not active 1 = Active

Notes on Channel Type KT

The permissible analog input modules can provide the analog value in four different notations. You must choose the one you want by setting the KT parameter accordingly.

KT = 3 : Absolute value between 4 and 20 mA
KT = 4 : Unipolar representation
KT = 5 : Bipolar absolute value
KT = 6 : Bipolar fixed-point number

If the KT parameter is set to a value less than 3, the function block uses KT = 4; if it is set to a value exceeding 6, the function block uses KT = 6. The KT parameter must be in agreement with the method of representation set on the module itself.

Notes on Output Value XA

Input value XE from the analog input module is converted according to the following formulas, depending on channel type parameter KT.

OGR : Upper limit value UGR : Lower limit value
XE : Input value XA : Output value

$$KT = 3 \quad XA = \frac{UGR \times (2560 - XE) + OGR \times (XE - 512)}{2048}$$

$$KT = 4 \quad XA = \frac{UGR \times (2048 - XE) + OGR \times XE}{2048}$$

$$KT = 5/6 \quad XA = \frac{UGR \times (2048 - XE) + OGR \times (XE + 2048)}{4096}$$

Notes on UGR/OGR

Range limits for the output value

The analog value can be represented as a physical value if suitable range limits are chosen.

Example:

Analog Value Range	Phys. Value	Range Limits		Resolution
0 to 10 V	2 to 150 °C	2000000 +01	1500000 +03	0.1 °C

Notes on "Range Violation" (BU) Bits

Range violation BU

In the case of modules which were configured as three-channel redundant modules, the upper and lower limit values specified with COM H are used to check for range violations. If the analog value from all three modules is in the overflow range (analog value > +4096 or < –4096 units), the "Range violation" bit is set and the analog value limited to +4096 or –4096 units.

Channel type 3: 4 to 20 mA The BU bit is also set when the value is in the range from 3 to 4 mA

Wirebreak

In the event of a wirebreak on all three modules, XA is set to 0 and the 'Wirebreak' bit set.

Notes on Selective Sampling

For selective sampling, the function block initiates encoding and sets activity bit TBIT to '1'; this bit remains at '1' until encoding has been completed.

Only one channel on a module may be sampled at any given time. An interlock must therefore be placed on the associated function block calls with the aid of the activity bit. All three modules are activated automatically.

When using selective sampling, all AI addresses must also be configured as AQs (type 20), and an AQ channel must be configured for the third AI channel.

Notes on Error Detection

The three analog values are checked for discrepancies. The permissible discrepancy value is calculated from an absolute value (in units corresponding to the specified digital notation for the analog value) to which a relative value (a percentage of the maximum value of the three analog values) is added.

$$D_{perm.} = ABS + \frac{REL \times RAWV}{100}$$

$D_{perm.}$:	Permissible analog value discrepancy
ABS:	Absolute component of the configured discrepancy
REL:	Relative component of the configured discrepancy
RAWV:	The middle value of the three current analog values

Example

The following was configured with COM 155H:

ABS: 100 REL: 10%

Actual analog value, subunit A: 1000

subunit B: 980

Actual analog value, 3rd channel: 1040

Actual discrepancy: D1 = 1000 – 980 = 20

D2 = 1040 – 1000 = 40

Permissible discrepancy $D_{perm.} = 100 + \frac{10 \times 1000}{100} = 200$

It follows that the actual discrepancy lies within the permissible range.

When FB 43:3-RLG AE is called, it reads the analog values from both subunits, exchanges them, and uses the middle value of the three analog values. If a one-sided, locatable error occurs (timeout, wirebreak), the module is passivated; that is, it is no longer accessed (1-out-of-2 operation).

If the 155H system program detects a discrepancy error, the module whose analog value differs most from the other two (2-out-of-3) is passivated when the configured discrepancy time has elapsed.

If a number of modules report different errors, passivation is prioritized as follows:

1. Timeout (QVZ)
2. Wirebreak
3. Overflow
4. Range violation.

If FB 43:3-RLG AE detects only a range violation or overflow condition for all three AI modules, the standardized analog value is passed together with the "Range violation" (BU) or "Overflow" (BU) bit.

If FB 43:3-RLG AE detects a "wirebreak" error in all three modules, it sets the "Wirebreak" (FB) bit.

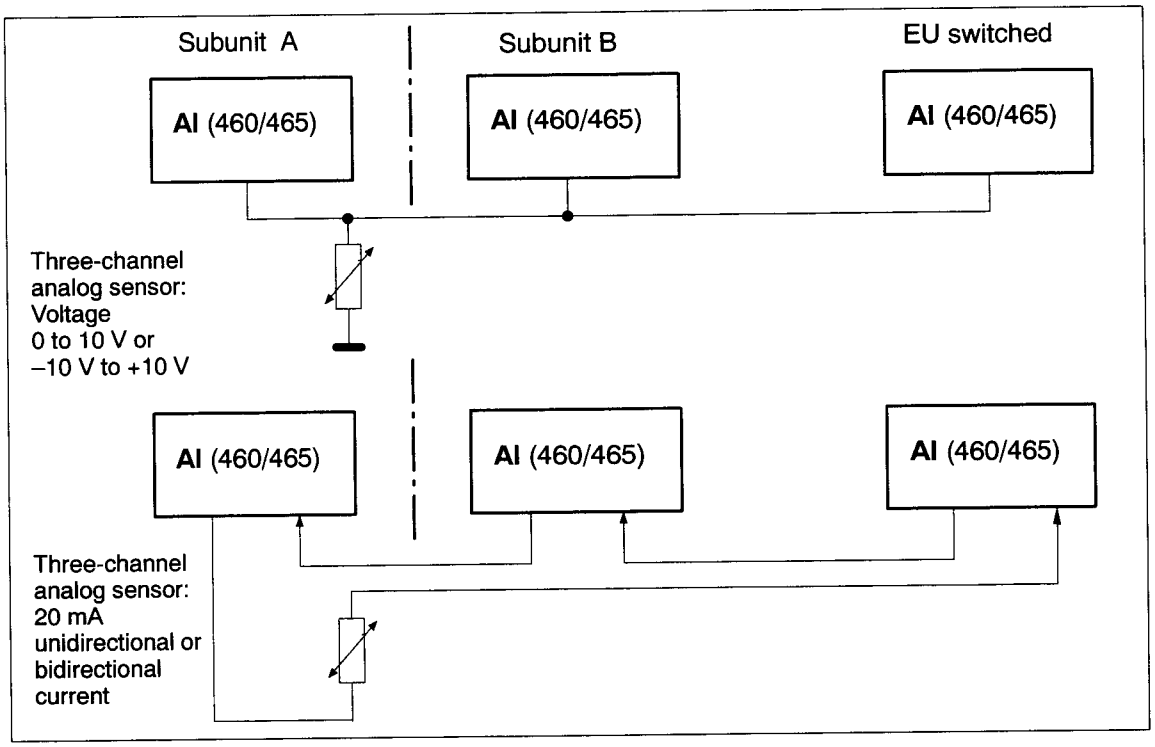


Figure 4-11 Three-Channel Redundant 1-out-of-3 AIs with One Sensor

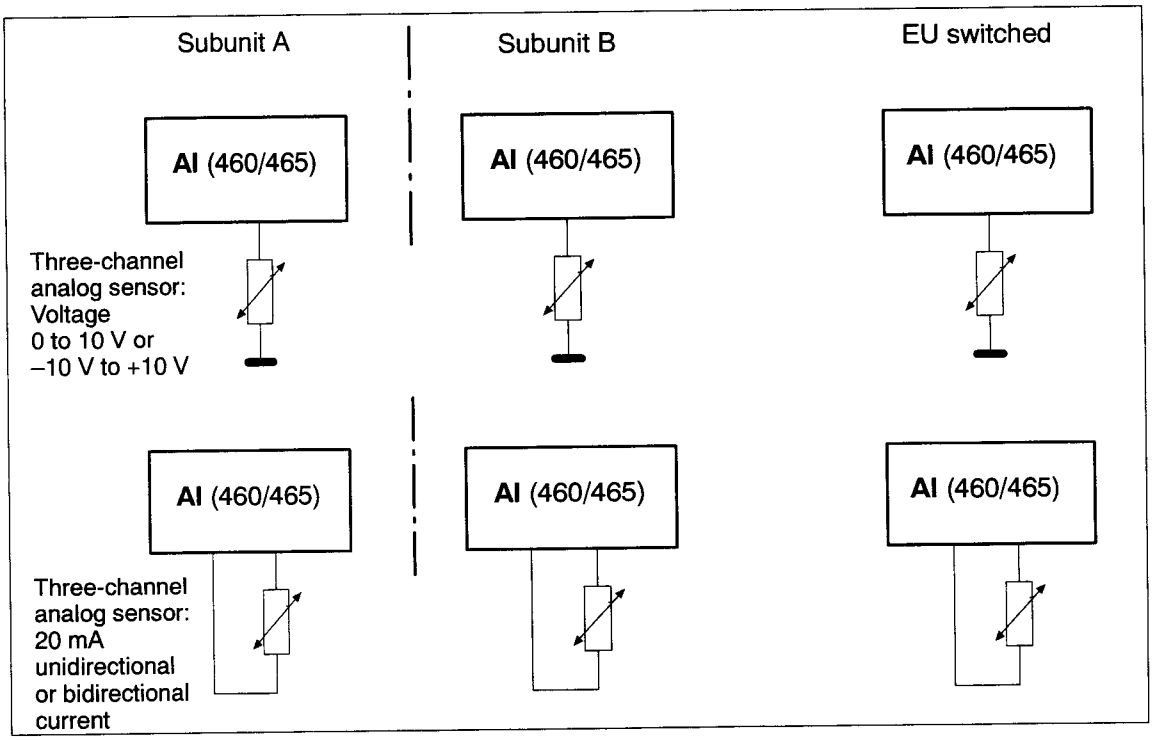


Figure 4-12 Three-Channel Redundant 1-out-of-3 AIs with Three Sensors

One-Sided Analog Outputs

All direct access operations (such as T PY, T OY and T OW), to one-sided analog outputs are allowed. Output to the I/Os is direct on two channels.

Redundant Analog Outputs

The following applies to AQs: If a STEP 5 operation (T PW, T OW) is used to output a value to a redundant channel, then the value is output to **both** subunits. Wiring and readback must be user-configured.

If error detection with error locating is required, you can make this possible by implementing your own readback analog inputs or selecting the AQ type 21 which is supported by the operating system.

Note: When analog modules are used, the BASP signal is ignored; that is, the last analog value is retained. Remember this when wiring the redundant analog outputs.

Redundant 2-Channel AQs without Error Locating Facility

The analog output (I/O type 20) has the same address in subunits A and B. No other switched or one-sided DQ or AQ may be configured under this address.

Redundant 2-Channel AQs with Error Locating Facility

The analog output (I/O type 21) has the same address in subunits A and B. All AQ channels are assumed to be intermittent, as no artificial value range test is executed. An active side change (relay switching every 10 hours) shows whether or not the passive AQ channel is still intact.

Fault Tolerance and NON-STOP Operation

Two-channel analog output modules may be used as NON-STOP AQs when the process is able to tolerate an AQ error output at regular intervals, which necessitates one or more FB call intervals. In the event of an error – on one side – the output to the actuator is switched, and the error thus controlled (Figure 4-13). A system with two-channel AQ will tolerate the following errors in one of the subunits:

- AQ module failure
- Failure of the rack power supply
- Failure of the load voltage in the event of redundant load voltage

The maximum duration (T_F) of an error output can be calculated as follows:

$T_F = T_D + T_{L-DQ} + T_Q$

TQ	Call interval for FB 41 (H:RLG:AA)
TD	Configured discrepancy time = AQ output delay + AI readback delay (rounded off to 10 ms units)
TL-DQ	Relay delay time of the L-DQ

Direct I/O Access

All direct two-address and four-address I/O access operations (such as T PW and TBGD) to two-channel AQs are permitted. When such an operation is executed, the operating system outputs the value to the two-channel AQ in **both** subunits. The readback AI is not checked. Address access operations, such as T PY, to two-channel AQs of type 21 are **not** permitted.

Subunit Failure

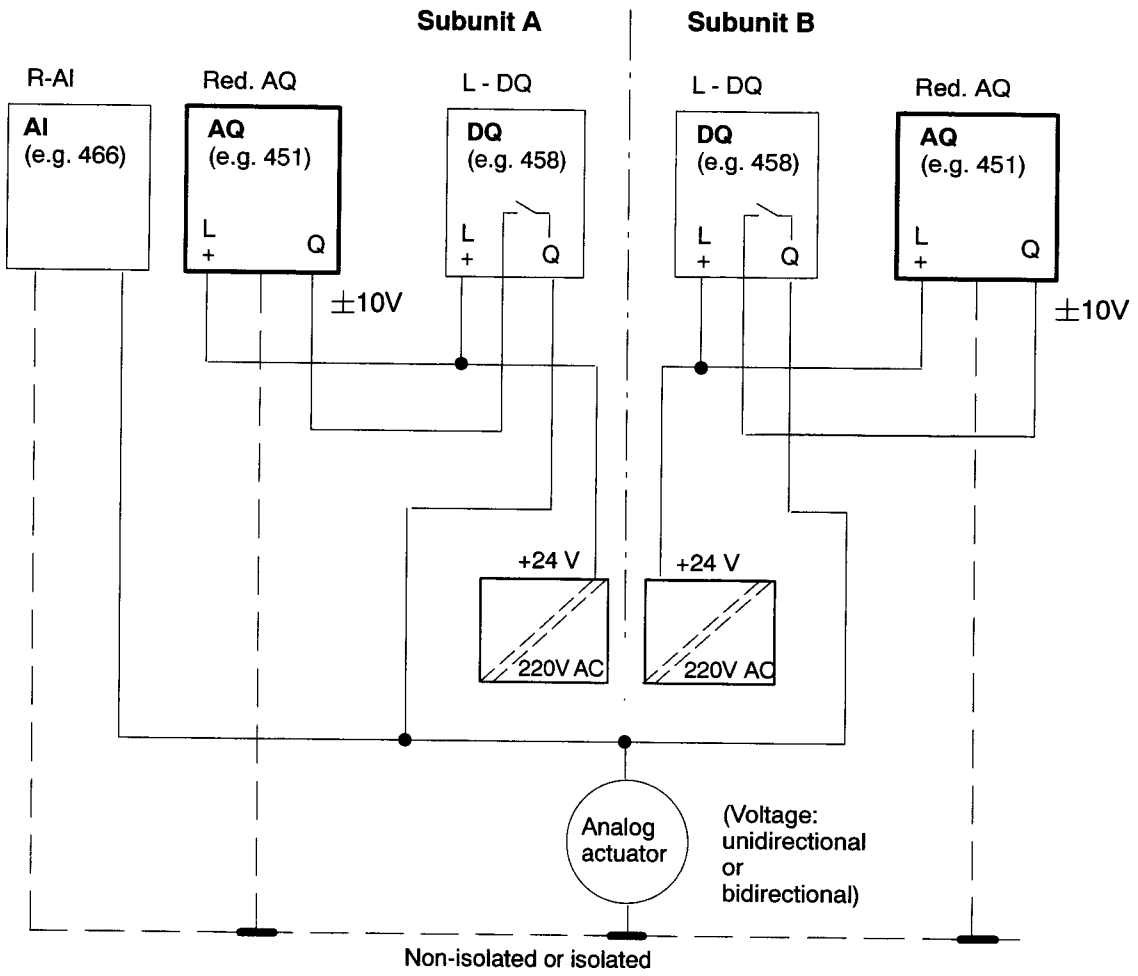
When a subunit fails, the operating system executes the switchover or decoupling routine, switching all L-DQ relay outputs in the intact subunit to "1" (see Figures 4-13 and 4-14).

Depassivation must be initiated following repairs to a two-channel AQ. When depassivation has been completed, the two-channel AQ is once again fault tolerant.

The setting of the voltage range for AQ and R-AI must match for voltage output.

For current output, the R-AI module must be set to 1 V (1.25 V for the 466).

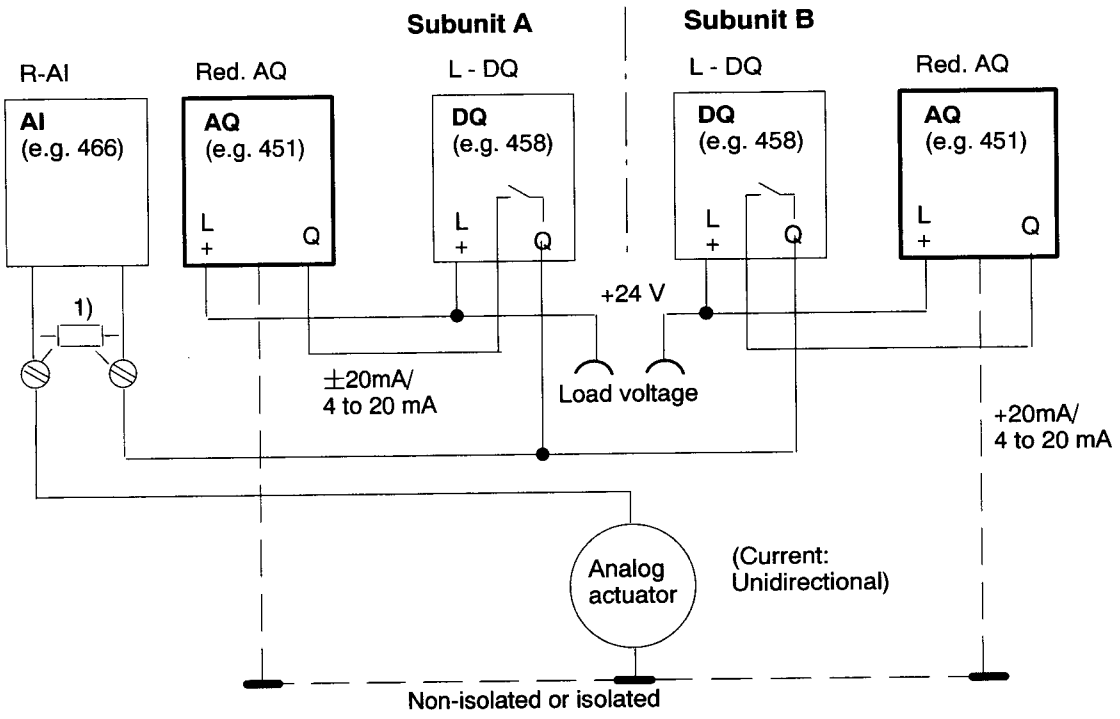
Only two's complement number representation may be used for the setting.



L-DQ : Locating DQ
Red. AQ : Redundant AQ
R-AI : Readback AI

Use of a load voltage supply is also permitted. However, if the load voltage fails, the entire redundant AQ group fails.

Figure 4-13 Redundant (1-out-of-2) AQs with Error Locating on Voltage Output (I/O Type 21)



1) 50 W , 0.1 %
Setting for the AI module to 1 V. If the 466 module is used as the readback AI, 1.25 V should be set.

Figure 4-14 Redundant (1-out-of-2) AQs with Error Locating on Current Output (I/O Type 21)

At the current input, the module type 6ES5 465-... is not permitted.

FB for 2-Channel Redundant AQs (FB 41)

Standard function block FB 41 "H-RLG:AA" is provided as the means to output two-channel analog values. FB 41 is part of the COM 155H package. The FB number may be changed on loading.

The function block converts an input value XE into an output value for an analog module on the basis of nominal range limits UGR and OGR. An error is reported in the event of a range violation. The function block also processes the system's response to the failure of a two-channel AQ component. Special configuring data must be entered with COM 155H for redundant AQs.

Execution Time of "Function Blocks"

The execution time of the "H" function block may exceed that of the standard function block, depending on the operations used:

FB 41 H-RLG:AA, for two-channel redundant AQs by approx.:	750 µs

Using FB 41

FB 41 does the following:

- Both subunits still functioning properly

  FB 41 reads the readback AI and compares the value read with the corresponding value in the PIQ. If the discrepancy time elapses, the L-DQ relay is switched and and H error ("Passivation") reported.

  The output value is output to the AQ module in both subunits and the raw value stored in the PIQ.

- One I/O side has failed (CPU in redundant mode)

  FB 41 reads the readback AI and compares the value read with the corresponding value in the PIQ. If the discrepancy time elapses, an H error ("Error") is reported. The readback AI is passivated.

  The output value is output to the AQ module and the raw value stored in the PIQ.

- One subunit is operating in Solo mode

  The output value is output to the AQ module and the raw value stored in the PIQ. The readback AI is not read.

Rules for the User Program

A) The channel to which the L-DQ bit is assigned must be the AQ for which FB 41 is called most often, so that the test switchover can run as quickly as possible. For example;

OB xx		10 ms
SEGMENT 1		0000
0000	: L SW xx	
0001	: T PW 128	Direct access = without error detection
0002	: L SW xy	
0003	: T PW 130	Direct access = without error detection

The output channels must be updated by the FB 41 (even if less often), as only the FB 41 executes the L-DQ switching, error detection and error localization.

```
OB 11                    10 ms
SEGMENT 1                0000
0000          :  JU FB 41
0001 NAME     :  H-RLG AA
0002 XE       :  FD 10      Input value XE (floating point)
0003 BG       :  KF + 128   Base address of the two-channel AQ
0004 P/Q      :  KS P       Area ID: P
0005KNKT      :  KY 0,0     Channel 0
...
```

```
OB 13                    100 ms
SEGMENT 1                0000
0000          :  JU FB 41
0001 NAME     :  H-RLG AA
0002 XE       :  FD 20      Input value XE (floating point)
0003 BG       :  KF + 128   Base address of the two-channel AQ
0004 P/Q      :  KS P       Area ID: P
0005KNKT      :  KY 1,0     Channel 1
...
```

The parameter value «Indic. no. of updates (1...10)» is calculated from the number of direct access operations and FB 41 calls for the chosen channel per readback delay time.

If the value calculated is greater than 10, an AI module type should be chosen which encodes quicker (for example, 463 or 466).

B) To reduce the switchover surge on the analog output when a subunit fails to a minimum time, the given instruction sequence in OB 1 and OB 37 must be implemented : (with x= H flag word)

```
OB 1
SEGMENT 1              0000
0000  : A F x.1        Redundant mode
0001  : S S 100.0      Any flag to signal failure of partner
...
```

```
OB 37
SEGMENT 1          0000
0000          :  AN F x.1          Solo mode
0001          :  A S 100.0         and previous redundant mode
0002          :  JC FB 41          FB 41 switches all L-DQ bytes to
                                   0FFH immediately
0003 XE       :  FD 0              Not evaluated here
0004 BG       :  KF + 0            0 means failure of partner CPU
0005 P/Q      :  KS P              Area ID: P
0006KNKT      :  KY 0,0            0,0
0007 OGR      :  KG 00000 + 0      0
0009 UGR      :  KG 00000 + 0      0
000A FEH      :  F 10.0            always 0
000B BU       :  F 10.0            always 0
...
```

Calling FB 41

STEP 5 program:

```
OB xx
SEGMENT 1
        : JU FB 41
NAME :  H-RLG AA
XE     : D  FD10      Input value XE (floating point)
BG     : KF  +128     Base address of the two-channel AQ
P/Q    : KS  P        Area ID: P/Q
KNKT   : KY  0,1      Channel number, channel type
OGR    : KG  10000+0  Upper limit of the output value
UGR    : KG  00000+0  Lower limit of the output value
FEH    : F  16.0      Error bit for UGR ≤ OGR
BU     : F  16.1      Error bit for XE < UGR or XE > OGR
```

Input/Output Parameters:

Name	Parameter Type	Data Type	Description	Remarks
XE	I	D	Address of the input value XE	Input value (floating point) in the range UGR to OGR
BG	D	KF	Module address	P/Q = P: BG = 128 to 240 P/Q = Q: BG = 0 to 240
P/Q	D	KS	I/O area	P/Q = P: P area P/Q = Q: O area
KNKT	D	KY	Channel number KN Channel type KT	KN = 0 to 7 KT = 0 unipolar KT = 1 unipolar
OGR	D	KG	Upper limit of the output value	–1701412+39 $\leq$ +1701412+39
UGR	D	KG	Lower limit of the output value	–1701412+39 $\leq$ +1701412+39
FEH	Q	BI	Bit address for error flag UGR $\leq$ OGR	
BU	Q	BI	Bit address for "Range violation" bit	0 = UGR $\leq$ XE $\leq$ OGR 1 = XE $\leq$ UGR or XE $\leq$ OGR

BU : If XE lies outside the range limits UGR and OGR, the bit BU is set and the last valid value output continues to be output by the active module.

4.5 One-Sided I/Os

Configuration and Functionality

A one-sided module is always assigned to one of the two subunits. If this subunit fails, the modules assigned to it also fail. This means that this configuration provides no greater fault tolerance than the S5-155U.

The modules can be plugged into either the central controller or an expansion unit.

When using one-sided I/Os, it is of no significance which subunit is master. Both subunits receive the current signal states. If a timeout (QVZ) occurs on a one-sided I/O, the relevant I/O byte is passivated.

Figure 4-15 shows the various one-sided configurations.

Note:
If a subunit fails, the one-sided I/Os in that subunit can no longer carry out their assigned functions.

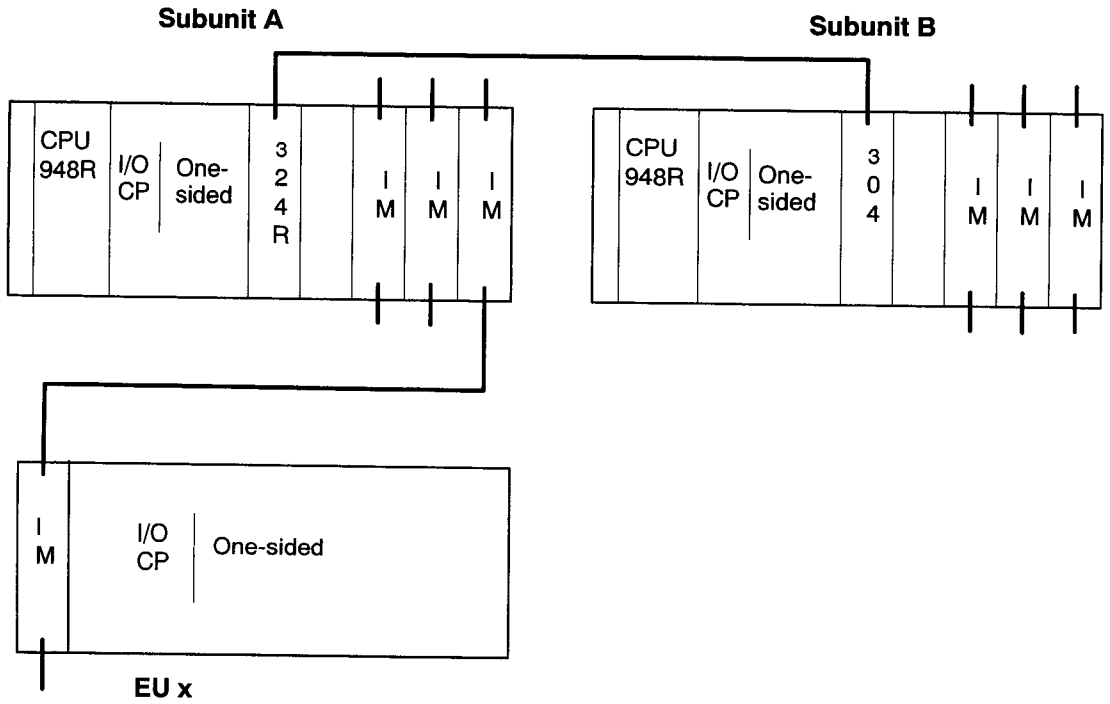


Figure 4-15 One-Sided I/O Operation and Permissible Modules

Interface Modules and Expansion Units (EUs)

The same interface modules and expansion units can be used for one-sided operation in the S5-155H as can be used in the S5-155U (refer to the S5-135U/155U System Manual, Chapter 4).

Digital and Analog Input/Output Modules

All I/O modules used in the S5-155U can be used for one-sided operation.

One-sided I/O modules should be used only for subprocesses whose failure can be tolerated in the event of a controller failure. The software for controlling these subprocesses should be implemented in blocks reserved for this purpose, and these blocks invoked conditionally; that is, only when the relevant subunit is running (see example). If this recommendation is not observed, a subunit failure will cause a continuous timeout (QVZ error), seriously affecting the scan time.

Sample program: One-sided I/Os assigned to subunit A

STL	Explanation
:O F X.1	Flag for "PLC in redundant mode"
:	Bit 2^1 in H flag word (high-order byte)
:O F X.4	Flag for "Central controller is subunit"
:	A", bit 2^4 in H flag word (high-order byte)
:JC FB Subproc.A	

The subprocess runs only when subunit A is running.

Proceed in the same manner as in the sample program above for one-sided I/Os assigned to subunit B.

Sample program: One-sided I/Os assigned to subunit B

STL	Explanation
:O F X.1	
:	
:ON F X.4	
:	
:JC FB Subproc.B	

Addressing

One-sided DIs or DQs may be assigned a given address once only, either in subunit A or subunit B, and the addresses used for one-sided DIs or DQs may not be used for switched or redundant DIs/DQs.

The same applies to AIs and AQs; that is, these modules may also be assigned a given address once only, and the addresses assigned to AIs/AQs may not be used for switched or redundant DIs/AIs or DQs/AQs. A one-sided third channel on a three-channel DI or AI, however, may be assigned the same address in the other subunit.

Address area:	One-sided DIs:	0FF000h to 0FF1FFh
	One-sided DQs:	0FF000h to 0FF1FFh
	One-sided AIs/AQs:	0FF080h to 0FF1FFh

All direct I/O access operations (e.g. T PY, T OY, T PW) are permitted for one-sided analog inputs/outputs.

4.6 Switched I/Os

Configuration and Functionality

The input or output module can be operated by either subunit. This provides enhanced fault tolerance compared to the S5-155U.

At least one 185U expansion unit is required to operate switched I/Os. The expansion unit is connected to both S5-155H central controllers via the IM 304/IM 314R interface modules. Up to 16 switched EUs can be operated in one S5-155H controller.

When a timeout (QVZ) occurs for a switched I/O, the byte is not passivated. On the first timeout for a switched I/O in redundant controller mode, a standby-master switchover is carried out and the "PLC fault" bit in the H flag word set.

Figure 4-16 illustrates the various switched I/O configurations.

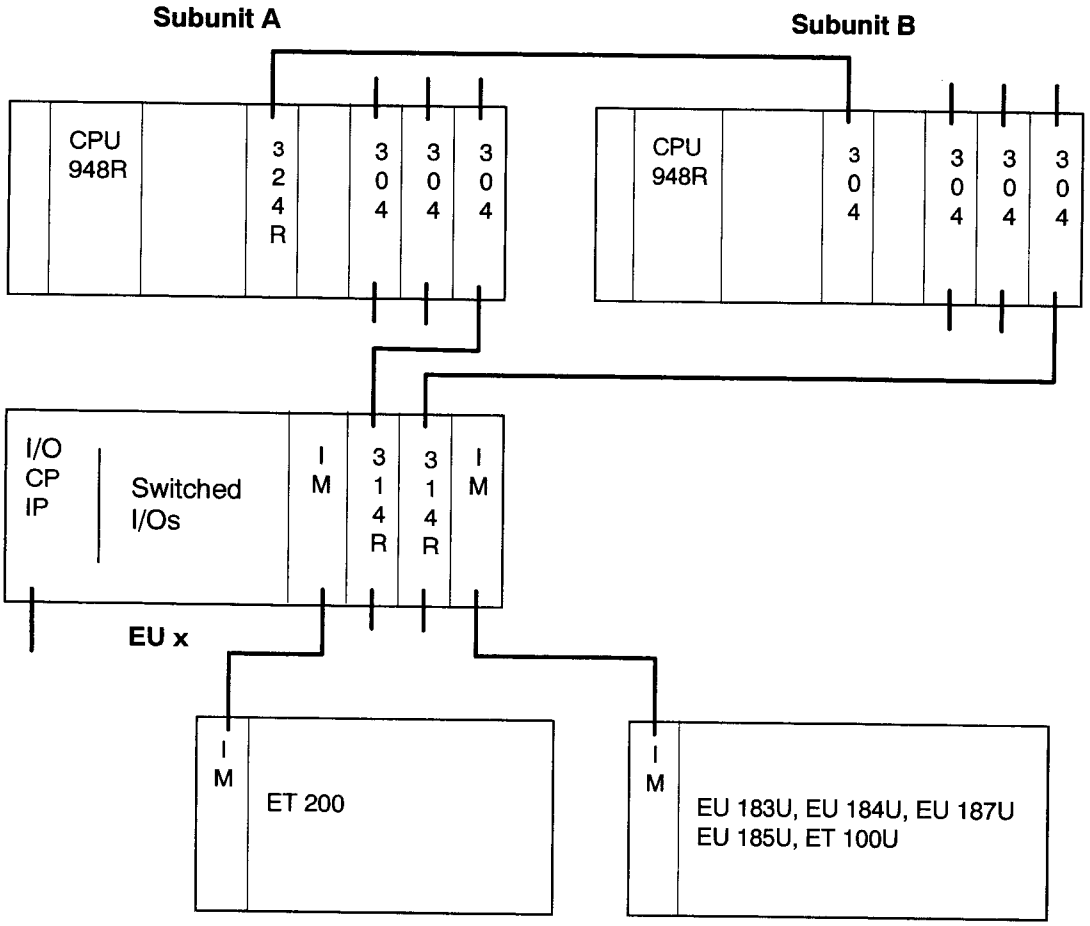


Figure 4-16 Switched I/Os and Permissible Modules

Interface Modules and Expansion Units (EUs)

At least one 185U expansion unit and the IM 304 to IM 314R interface modules are required to operate switched I/Os.

Note

If the IM 300 or IM 308 interface module is plugged into slot 163 in the expansion unit, the 183U, 184U and 187U or 185U and the ET 100U can also be operated.

In switched I/O configurations and a maximum **ET 100U** configuration, a time loop longer than the ET 100U configuration run (max. 4 s) must be programmed in OB 22 in both the master and standby controller.

Digital and Analog I/O Modules

All I/O modules which can be used in the S5-155U can also be used in switched I/O configurations.

Address area:	Switched DIs/DQs:	0FF000h to 0FF1FFh
		0FF300h to 0FF3FFh
		0FFC00h to 0FFDFFh
	Switched AIs/AQs:	0FF080h to 0FF1FFh
		0FF300h to 0FF3FFh
		0FFC00h to 0FFDFFh

All direct I/O access operations (T PY, T OY, and T PW, for example) are permitted for use with switched analog inputs/outputs.

4.7 Hybrid I/O Configurations

All three I/O modes (one-sided, switched and redundant) can be combined in one S5-155H.

The figure below shows possible hybrid configurations.

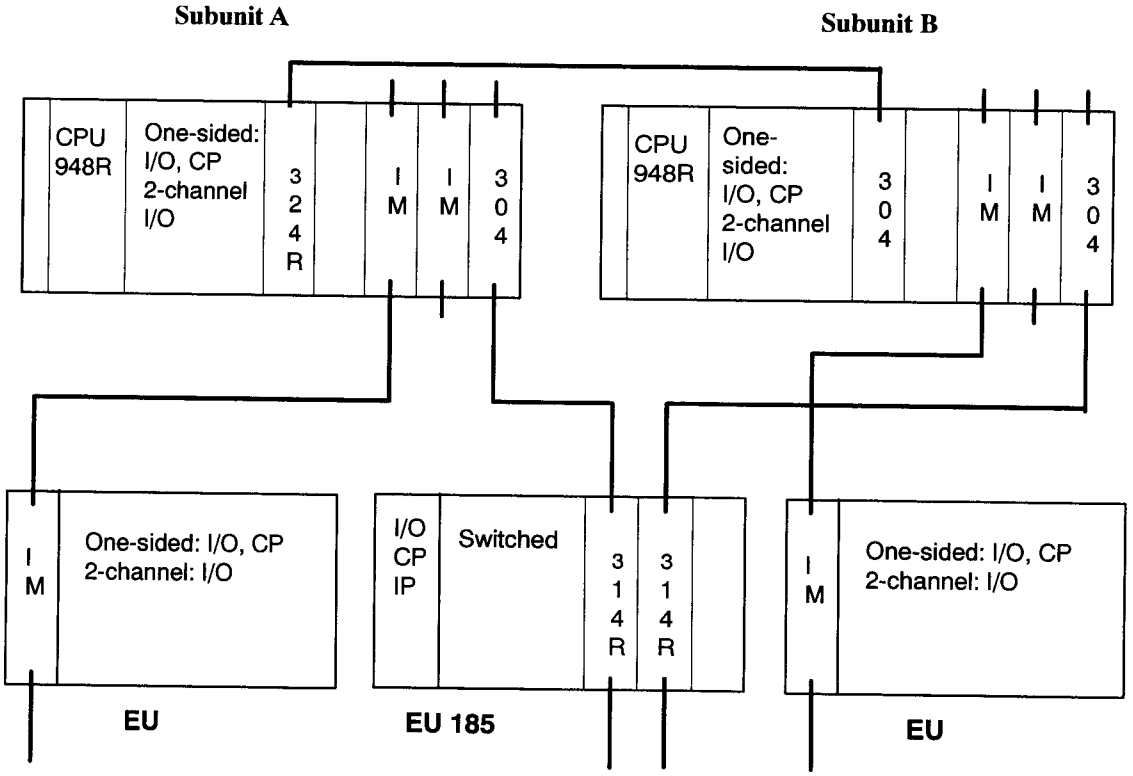


Figure 4-17 Hybrid I/O Configurations

Operation of Communications Processors and Intelligent I/Os in the S5-155H

5

This chapter discusses the use of intelligent I/Os (IPs) and communications processors (CPs) in the S5-155H programmable controller in one-sided, switched and redundant configurations. It also covers the data handling blocks (DHBs) for the S5-155H and when and how to invoke them in the Restart routine and in the cyclic program.

5.1 Intelligent I/Os in the S5-155H

Communications

Normally, the programmable controller communicates with its intelligent I/O modules via data handling blocks (DHBs). In some cases, communication between intelligent I/Os and the user program is supported by specific standard function blocks (FBs). These FBs use the DHBs exclusively for communication between the PLC and the IPs/CPs.

In the case of other intelligent I/Os, communication between the programmable controller and the intelligent I/Os and/or communications processors is implemented by module-specific standard FBs.

The data handling blocks (DHBs) for the S5-155H are on the COM 155H diskette.

5.2 One-Sided CP/IP Configurations

Communications Processors (CPs/IPs)

In a one-sided configuration, the degree of fault tolerance is no higher than that of an S5-155H. Such a configuration may therefore be used only when a failure can be tolerated.

In one-sided CP/IP configurations, you need only configure the interface number and the module-to-subunit assignments (refer to the description of COM 155H in this manual).

Interprocessor communication flags (for CPs) may not be used in one-sided configurations.

Incoming data is forwarded to both subunits, regardless of whether the subunit containing the CP/IP is the master or the standby.

Note

One-sided CPs should be used only for subprocesses which can be sacrificed completely in the event of a PLC failure. The software for controlling these subprocesses should be implemented in separate blocks which can be invoked **conditionally** (only when the relevant PLC is functional; see example). Failure to observe this recommendation will result in a continuous timeout should a PLC fail, which in turn will greatly increase the scan time.

Sample Program

One-sided CPs assigned to subunit A:

STL	Explanation
:O F X.1 :	"PLC in redundant mode" flag, bit 2^1 in the H flag word (high-order byte)
:O F X.4 :	Flag for "Central controller is subunit A", bit 2^4 in the H flag word (high-order byte)
:JC FB Subproc. A	

The subprocess is then controlled only when subunit A is operating.

One-sided CPs assigned to subunit B:

STL	Explanation
:O F X.1 :	
:ON F X.4 :	
:JC FB Subproc.B	

The subprocess is controlled only when subunit B is operating.

5.3 Switched CP/IP Configurations

CPs/IPs

The standby-master switchover is carried out without loss of data, regardless of whether the CPs/IPs are controlled by H data handling blocks or special function blocks.

All data which the master's CPU reads from the CPs/IPs is also forwarded to the standby CPU. Only the master CPU, however, writes to the CPs/IPs; the standby CPU suppresses write operations.

Interprocessor Communication Flags

Interprocessor communication flags (address range FF200 to FF2FF) are supported only in switched CP/IP mode.

These flags must be configured in data block DB 1. All other settings in DB 1 are irrelevant in this case.

The interprocessor communication output flags are output by the master controller, while the interprocessor communication input flags are passed from the master to the standby controller. Direct I/O access to "switched I/O" IPFs is thus permissible.

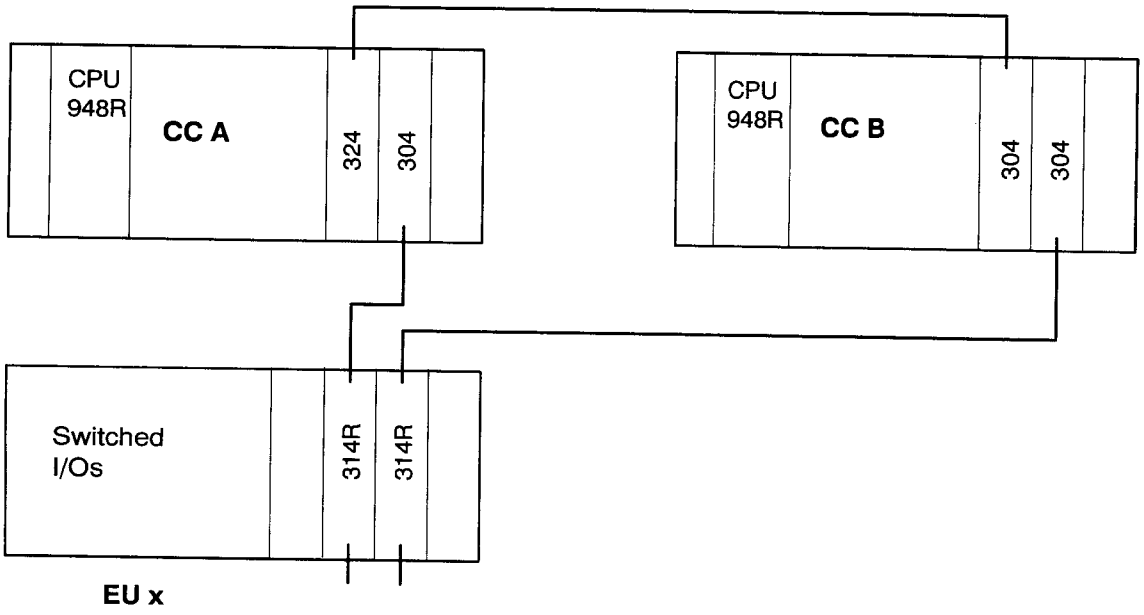


Figure 5-1 Switched I/O Configuration

5.4 Redundant Communications Processor (CP) Configurations

Configurations with Communications Processors

There are four configurations for redundant communications processors:

a) Two-channel redundant CP configuration (Figure 5-2)
b) Switched redundant CP configuration (Figure 5-3)

A single or double bus may be used!

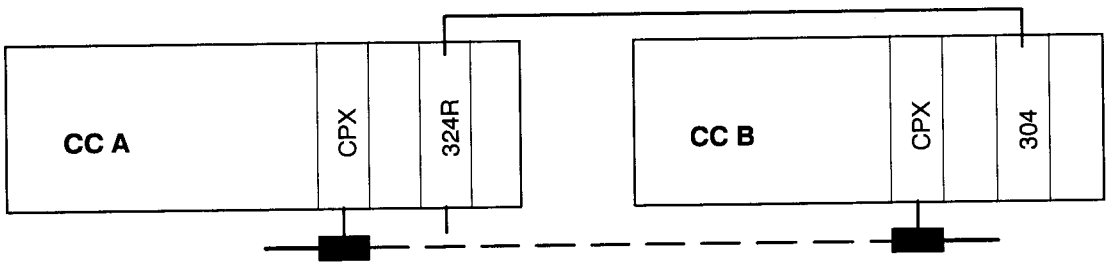


Figure 5-2 Two-Channel Redundant CP Configurations (Variant a)

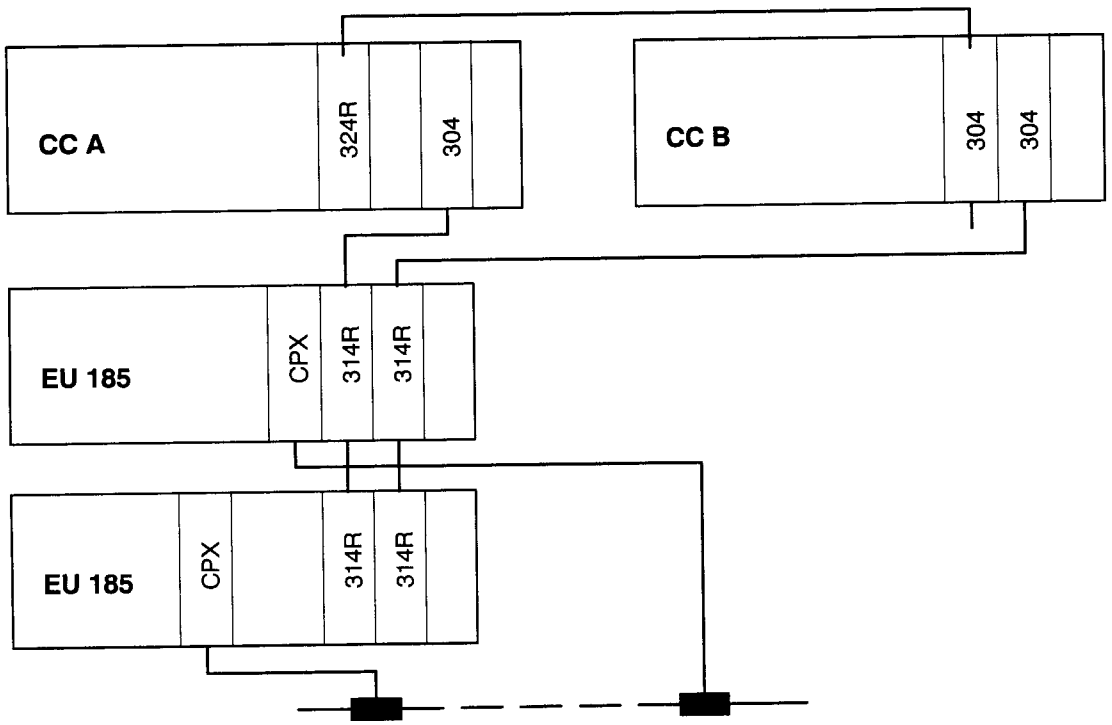


Figure 5-3 Switched Redundant CP Configurations (Variant b)

CP/IP Redundancy

CPs/IPs can also be redundant. They can be plugged into switched EUs, which is then called a "switched redundant" configuration. The CPs can also be plugged into the two subunits, which is then called a "two-channel redundant" configuration.

Selection criteria:

The best configuration for a given situation depends on the application.

Advantages of a switched redundant configuration:

- The CPs/IPs remain redundant even when a central controller fails
- The central controllers remain redundant even when a CP/IP fails
- Shorter scan time

Disadvantages of a switched redundant configuration:

- At least two switched EUs are required.
- While a CP/IP is being repaired, the EU into which it was plugged must be shut down. This means that all other I/O modules in that expansion unit are shut down as well.

Advantages of a two-channel redundant configuration:

- No switched EUs are required.
- CP repairs usually require that only redundant components be separated from the power supply.

Disadvantages of a two-channel redundant configuration:

- A significant increase in the scan time.

When programming, the two CPs must be regarded as independent modules. How redundancy is handled depends on the required functionality, and must be programmed by the user.

Switched Redundant CPs/IPs

If a CP/IP is to be plugged into an EU, you will need at least two EUs. A CP or IP must be plugged into each EU. When data arrives at a CP/IP, it is automatically forwarded to the other subunit.

Two-Channel Redundant CPs/IPs

In order to implement a two-channel redundant CP configuration, one CP must be plugged into each subunit. The two CPs reserve different pages, and operate independently of one another. If data arrives at one CP, it is automatically forwarded to the other subunit.

The user must program the redundancy function for a switched or two-channel redundant configuration himself. The user program must be written so as to stipulate which CP/IP is active, and must be able to detect a CP/IP fault and switch, if necessary, to the other CP/IP. The operating system ensures that the data is the same in both subunits. The two CPs/IPs must be regarded as separate, independent modules.

In contrast to redundant I/O modules, redundant CPs and IPs reserve different addresses or pages in each subunit.

Installation and Startup

6

This chapter uses examples to explain the procedures to follow when configuring and programming your programmable controller, as well as the procedures to follow to put it into operation.

Sections 6.1 to 6.5 build on one another and should therefore be read consecutively; that is, you should read the entire section even if you want to use only redundant I/Os.

The COM 155H User's Guide is also relevant to all points in this section.

6.1 Installing the S5-155H

General Remarks

The SIMATIC S5 installation guidelines apply.

To avoid confusion, the redundant central controllers should be installed in separate cabinets. This has the following advantages:

- Two redundant power supply units
- Less confusion when laying the cables
- Better visual monitoring of master and standby

Example of an Installation

S5-155H with four switched EUs:

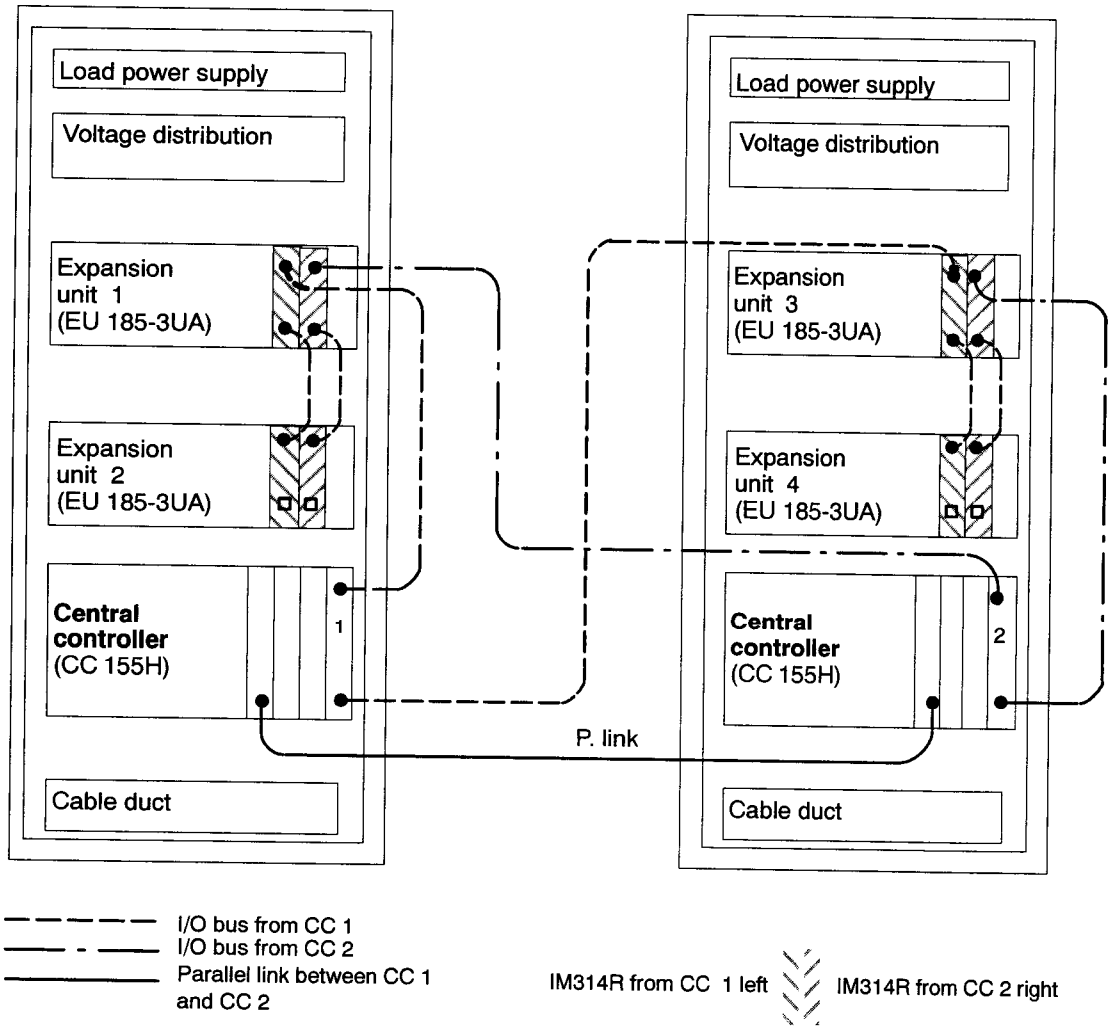


Figure 6-1 Installing an S5-155H

Recommendations and Rules for Hardware Installation

- The **central controllers'** power supply units should be fed from two separate and independent circuits.
- The power supply for the **expansion units** must have a fault-tolerant infeed (e.g. 24 V with backup batteries).

The figure below shows a possible solution:

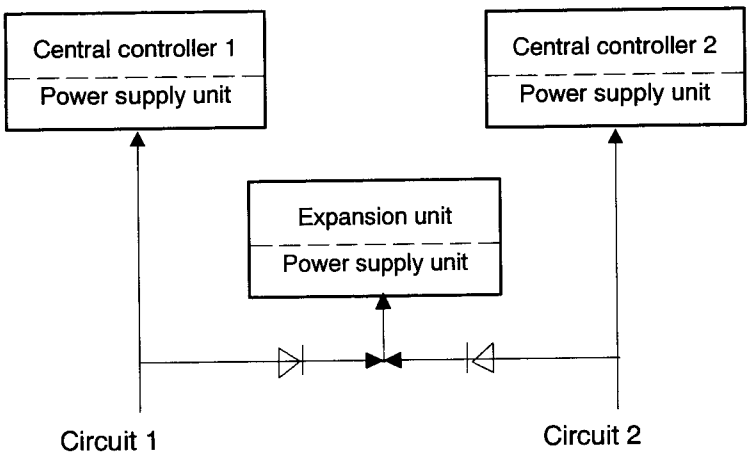


Figure 6-2 Infeed to the Power Supply Units

- It is advisable to have separate load power supplies connected to separate and independent circuits in each cabinet. They should be coupled to each other via diodes in such a way that, if one of the power supply units should fail, the other would take over. Load rating, fusing and conductor cross-section must be selected accordingly.
- If you have to run considerable lengths of connecting cables outside the cabinet, make sure the redundant cables are in separate cable ducts. This enhances fault tolerance.
- Connect all central controllers, expansion units and cabinets to an equipotential bonding conductor with a cross-sectional area of at least 10 mm^2.
- The difference in potential when linking up with cable type 721 must be < 7 V.
- To improve noise immunity, the shield of the type 721 connecting cable for the I/O buses of other EUs in a neighboring cabinet and that of the 721 cable for the IM 304/IM 314R parallel interface should be connected to the cabinet via a shield bar.
- Also make sure that the I/O buses are uniquely assigned to the central controllers and interface modules. For example:

  Central controller 1 = left interface module IM 314R

  Central controller 2 = right interface module IM 314R

Please refer to the "Installation Guidelines" in the S5-135U/155U System Manual for additional recommendations.

955 Power Supply Block

Note

You must insert a jumper in the two 955 power supply blocks for the 135U/155U central controller to deactivate the 24 V watchdog. To do so, loosen the two screws in the power supply block, remove the entire power supply unit, and insert jumper BA-EX.

Jumper BA-EX must be open in the 955 power supplies for the expansion units. This corresponds to the factory setting.

Failure and Repair of One-Sided/ Redundant Expansion Unit

When using the IM 304/IM 314-3U... link and several expansion units in series, note that if the first expansion unit in the phase fails, all other expansion units in the phase will also fail. To increase the fault tolerance, only one expansion unit should be used per phase.

6.2 Configuring the IM 304/IM 324R Parallel Link

Starting Point

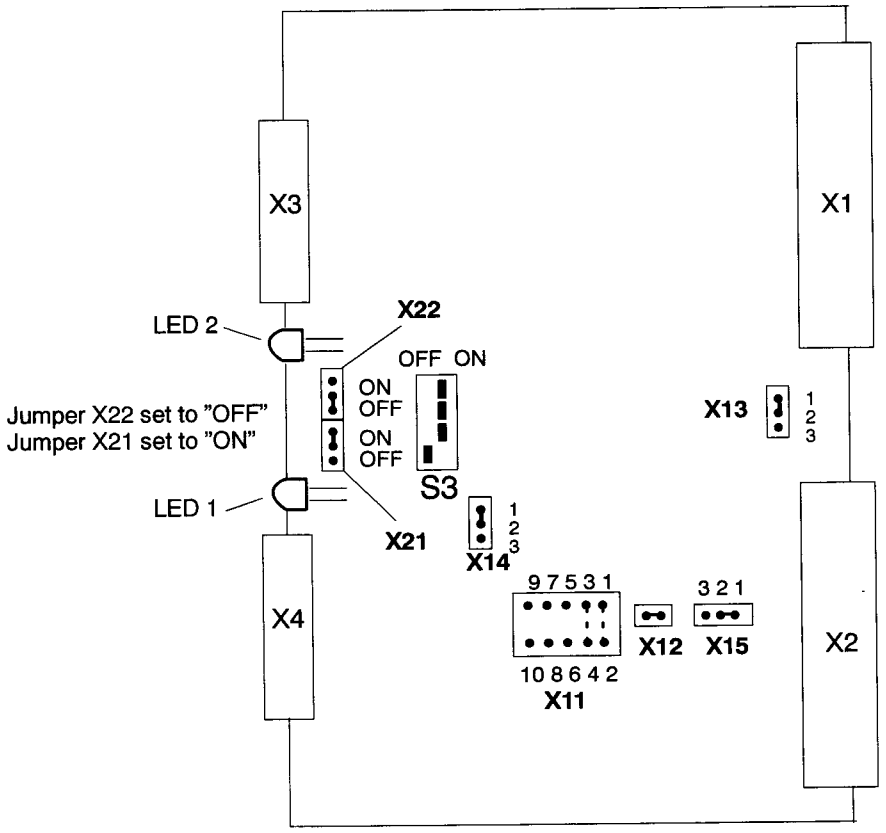
It has been assumed that a CPU 948R on which an overall reset has been performed is plugged into each subunit, and that the mains power is switched off.

Note carefully: Before interconnecting the two subunits via the IM 304/ IM 324R interface modules, check the jumper settings on the IM 304.

Jumper Settings on the IM 304

Jumper settings on the 6ES5304-3UB... module for a CC-to-CC parallel link via the IM 304 and IM 324R interface modules.

Caution: The IM 304 contains electrostatically sensitive components!



X11: Adaption to different cable lengths

	Jumper plug X11				
Jumper position	9 7 5 3 1 *) 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2
Cable length	Max. 10 m	10 to 100 m			

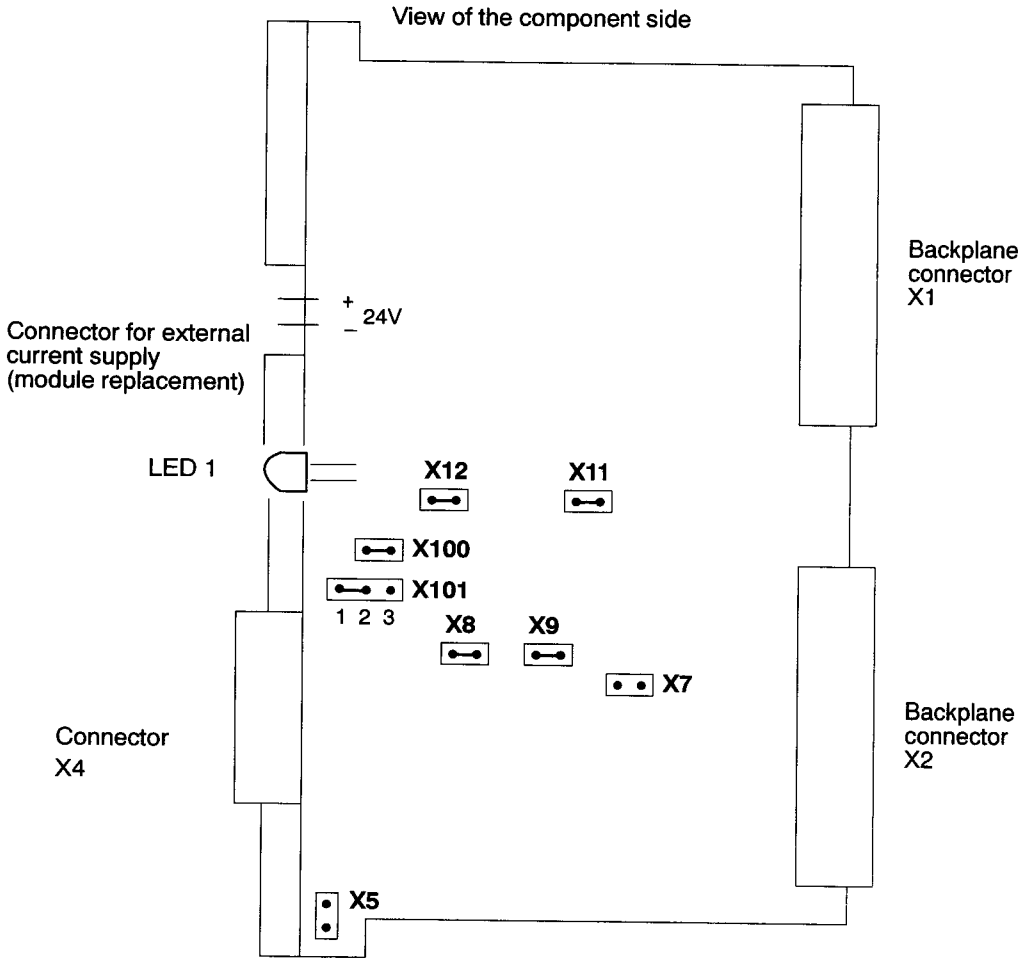
*) This setting is permissible only for the IM 304 - IM 324R link in the S5-155H. The length of the link to interface X4 determines the position of jumper X11.

Figure 6-3 Jumper Settings on the 6ES5 304-3UB** Module

IM 324R Interface Module

The IM 324R is required to establish a symmetrical 16-bit wide address/data link to a common dual-port RAM. The frontplate (lower half) is equipped with a connector for the symmetrical cable. Another connector on the top half of the frontplate is for changing modules during operation. It is used to connect an external voltage source with safe electrical isolation to VDE0160. The voltage range and current range data can be found under Technical Specifications (Section 11.2).

Caution! The IM 324R contains electrostatically sensitive components!



X101/1-2: The jumper configuration must include this jumper (distinguishes between an S5-115H and an S5-155H)

Figure 6-4 IM 324R Jumper Settings (Factory Settings)

1. Plug the IM 304/IM 324R modules into slot 131 in each subunit and connect them with a 6ES5 721-xxx cable.
2. Put the two PLCs into operation successively:

   Refer to Section 3.2 ("CPU 948R Installation and Startup Procedures").

   The green LED on the IM 324R must show a steady light (if this is **not** the case, then the IM 324R is defective).

   This allows you to operate the S5-155H in its minimum configuration:

   One subunit enters the RUN mode as master upon completion of the self-test; the RUN LED shows a steady light. The RUN LED on the other subunit flashes, showing that it is the standby.

6.3 Configuring One-Sided I/Os and Putting Them Into Operation

Starting Point

The two subunits are connected to one another via the parallel link. The CPUs have been reset (overall reset). The mains power is switched off.

- Plug the input, output or CP module you want to operate in one-sided mode into either subunit A or B or into an expansion unit assigned to either subunit A or B.
- If you plug the module into an expansion unit, be sure to use a suitable interface module.

Initializing One-Sided I/Os (COM 155H)

Switch on your programmer and start the COM 155H software (on floppy or hard disk) via the S5 command interpreter or the main menu.

Proceed as follows to configure your one-sided I/Os:

1. Choose between digital and/or analog inputs/outputs or CPs in the "I/O Configuring" form (refer to Chapter 4 of the COM 155H User's Guide).

   Complete the screen form for each input or output byte or word and for each CP interface.

   For digital or analog inputs/outputs, first set the byte number (invoke the "SEARCH" function and enter the byte number), then enter the relevant type number.

   Only the following types may be specified for one-sided I/Os:

   - DI: Type 1
   - DQ: Type 8
   - AI: Type 13
   - AQ: Type 18

   You must also specify the subunit (A or B).

2. For a CP in a one-sided configuration, enter the interface number first, then the type number. In this case, you would enter CP Type 24.

   Specify the subunit to which the CP is assigned.

3. Press <RETURN> after each entry.
4. Connect your programmer to subunit A or B (Power ON). The CPUs' mode selectors are set to "STOP".
5. You must now transfer your configuring data to the PLC.

Putting One-Sided I/Os into Operation

Proceed as follows:

1. Set the mode selector switch of the subunit to which the one-sided I/Os are assigned to RUN.
2. Select "H-ERROR" in COM 155H's basic "DIAGNOSTICS" form. Read any entries you find there, and rectify the problems with the aid of your manual.
3. When all errors have been rectified, set the subunit back to STOP.
4. Now set the other subunit to RUN and proceed as described above. When you have once again rectified all errors, switch both CPUs to STOP and load your STEP 5 program into one of the subunits.
5. Now, configure the DB/DX numbers for your STEP 5 program and transfer DX1 to one of the subunits.
6. Execute a cold restart on that subunit.
   Following completion of the self-test (RUN LED and STOP LED both show a steady light), this subunit will enter the RUN mode as master (the RUN LED shows a steady light).
7. Execute a cold restart on the other subunit. The subunit links up to the master, receives the user program from the master, and, after completing the self-test and going through the updating procedures, it enters the RUN mode as standby (the RUN LED flashes).

6.4 Configuring Switched I/Os and Putting Them Into Operation

Starting Point

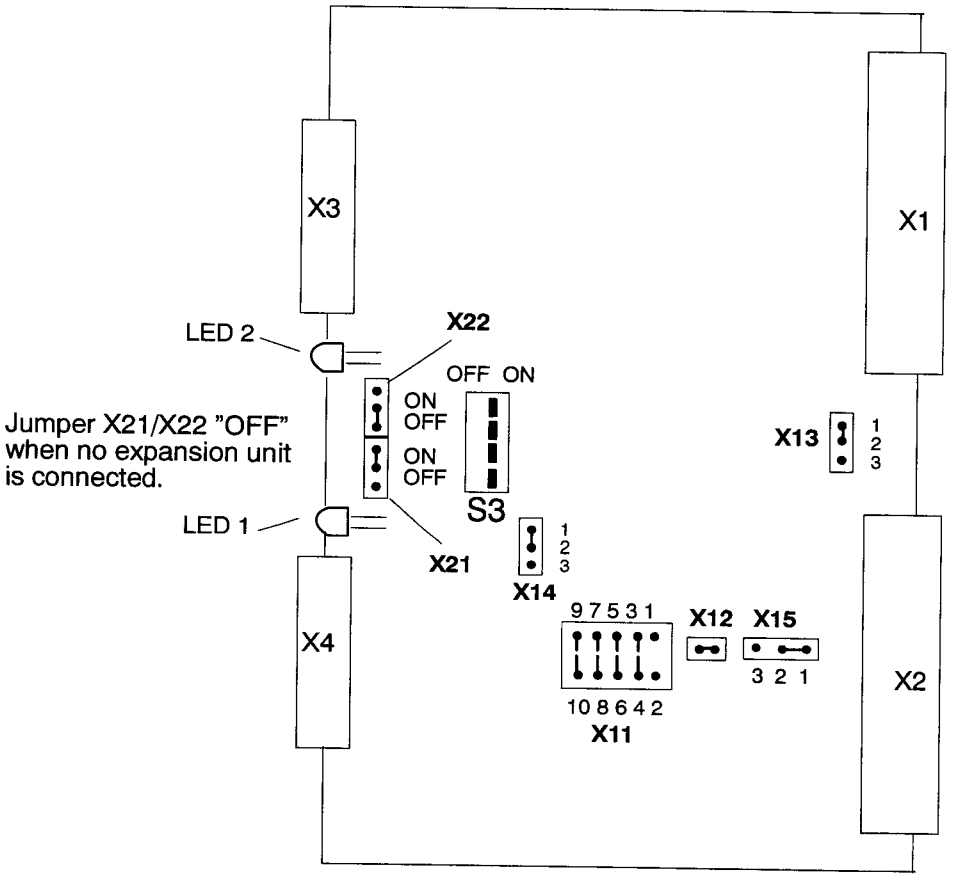
An I/O module in an EU 185 is to be operated as a switched I/O (an I/O which can be switched from bus to bus).

The expansion unit is connected symmetrically to the two subunits via an IM 304/IM 314R interface module.

1. First, check the jumper settings on the modules.
2. Plug an IM 304 into one of the last four slots in subunit A and subunit B.
3. Plug an IM 314R into slots 145 and 156 of the EU 185.

Jumper Settings on the IM 304

Jumper settings on a 6ES5304-3UB... module with a symmetrical CC-EU link via IM 304/IM 314R.



X11: Adaptation to different cable lengths

	Jumper plug X11				
Jumper position	9 7 5 3 1 *) 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2
Cable length		1 to 100 m	100 to 250 m	250 to 450 m	450 to 600 m

*) These settings are permissible for the IM 304 - IM 324R link in the S5-155H only. The longest link to interface X3 or X4 determines the position of jumper X11.

Figure 6-5 Jumper Settings on the 6ES5 304-3UB... Module

6

IM 314R Interface Module

Caution! The IM 314R contains electrostatically sensitive components!

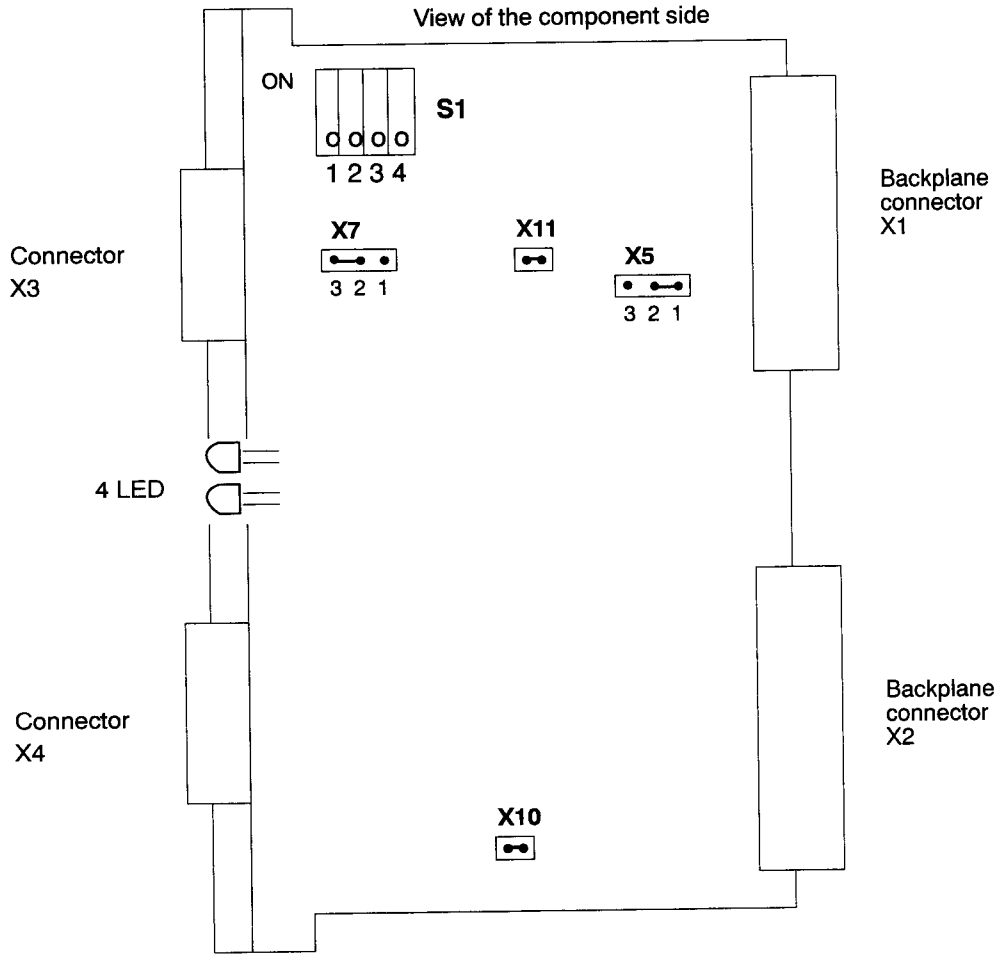


Figure 6-6 Jumper Settings on the IM 314R

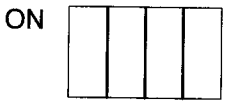
X3: Connector for the 721 cable coming directly from the subunit's IM 304 or, if several EUs are used, from the preceding IM 314R.

X4: Connector for the 721 cable to the next IM 314R or, if this is the last IM 314R on the bus, for the 760-0HA11 terminating-resistor connector.

Interfaces X3 and X4 are galvanically linked; even if the power supply to the EU 185 containing the IM 314R should fail, the bus connection remains fully intact.

Setting the EU Number

The number of the expansion unit must be set on switch S1, whereby the same EU number must be set for both IM 314Rs (you must specify a block number (see below) for this EU number during your I/O configuring session with COM 155H).



x = Switch ON
o = Switch OFF

1	2	3	4	EU No.:
o	o	o	o	00
x	o	o	o	01
o	x	o	o	02
x	x	o	o	03
o	o	x	o	04
x	o	x	o	05
o	x	x	o	06
x	x	x	o	07
o	o	o	x	08
x	o	o	x	09
o	x	o	x	10
x	x	o	x	11
o	o	x	x	12
x	o	x	x	13
o	x	x	x	14
x	x	x	x	15

LEDs on the Frontplate

The IM 314R's frontplate is equipped with four LEDs:

- The green "F" LED lights to indicate "Master" status
- The red "T" LED lights to indicate "Test"
- The red "BF" LED lights when there is a malfunction, or when the CC stops
- The yellow "R" LED lights to indicate "Standby" and "Ready to take control"

Table 6-1 LEDs on the IM 314R Frontplate

F	T	R	BF	Operational status
–	–	–	ON	CC at STOP or failed or standby is restarting
ON	ON	–	–	Master is restarting
ON	–	–	–	Master is in RUN mode
–	–	ON	–	Standby is in RUN mode
–	ON	–	–	EU not configured and associated CC is in RUN mode
–	–	–	–	Not configured and EU power OFF/ON

Command Output Disable (BASP) Signal

The IM 314R disables the expansion unit's digital outputs (BASP signal) in the following situations:

- When both subunits generate a BASP signal
- When the IM 314R is in "Test" mode
- When both subunits are switch off or when both of an EU's I/O buses are defective

The IM 314R cannot access the EU bus as long as the EU BASP signal is present.

- Plug both IM 314Rs into the EU (slots 145 and 156). Insert the 721 cable from the subunit's IM 304 into the upper front connector X3 on the IM 314R. Connector X4 is for the 760-0HA11 terminating resistor.

Initializing Switched I/Os (COM H)

The following parameters are required to configure your switched I/Os:

1. In COM 155H's "IM 314R" form, specify the I/O area for your EU numbers (must be identical to the setting on the IM 314R!).

   This is done by entering a block number. Example:

   "I/O area for EU No. 5: O Address area FF000H to FF0FFH"

2. Press <RETURN> after each entry.

3. Specify whether you want to operate digital and/or analog inputs/outputs or CPs/IPs in Switched mode by making the appropriate entry in COM 155H's basic "Configuration of the I/Os" form.

   Fill out a form for each input or output byte or word and for each CP/IP interface.

   For digital or analog inputs/outputs, first set the byte number ("invoke the "SEARCH" function, then enter the byte number), then enter the type number.

The following types may be used as switched I/Os:

- DI Type 2
- DQ Type 9
- AI Type 14
- AQ Type 19

4. For a switched CP or IP, first enter the interface number, then the type number. In this case, that would be CP/IP Type 25.
5. Press <RETURN> after each entry.
6. Transfer your configuring data to the PLC as described in the section dealing with one-sided I/Os (COM 155H "System Handling" form).

Putting Switched I/Os into Operation

The programmer must be interfaced to either subunit A or B (power ON), and the mode selectors on both CPUs must be set to "STOP". Proceed as follows:

1. Set subunit A to RUN.
2. Load the error DB from the PLC into the programmer and read any entries you find there (screen COM 155H's basic "DIAGNOSTICS" form, then call the "H ERRORS" submenu). Rectify any errors you find with the help of your manual.
3. Switch subunit A back to STOP.
4. Repeat for subunit B.
5. Now configure the DB/DX numbers in your STEP 5 program using COM 155H's "TRAFDAT" form.
6. When all errors have been rectified, load your STEP 5 program into one of the subunits.
7. After you have loaded the complete DX 1 configuring data block into the PLC, execute a cold restart on both subunits. Subunits A and B will then enter the RUN mode, one as master, the other as standby (the master's RUN LED will show a steady light, the standby's RUN LED will flash).

6.5 Configuring Redundant I/Os and Putting Them Into Operation

Starting Point

You must have two of every module you want to operate in a redundant configuration. One of the modules is plugged into a permissible slot in subunit A, or into an EU of subunit A, the other into subunit B, or into an EU of subunit B. The two modules must have the same address in both subunits.

The subunits are connected to each other via an IM 304/IM 324R parallel link. Each subunit has its own CPU 948R, both of which are reset (overall reset). Both subunits are at POWER OFF, and both are equipped with the same I/O module.

Initializing Redundant I/Os (COM H)

The following parameters are required to configure your redundant I/Os:

1. You must make the following entries in COM H's "Initialize operating system" form:
   - Standard discrepancy time
   - Readback delay
   - IB 0 as redundant interrupt DI byte (yes or no)
2. When you have entered all parameters, press <RETURN>.
3. Specify in COM 155H's basic "Configuration of the I/Os" form whether you want to configure digital or analog I/Os or CPs.
4. Fill in a form for each redundant input or output and for each redundant CP:

   For digital or analog inputs/outputs, specify the byte number first ("SEARCH" function: Enter byte number), then enter the Type number.

   Only the following types can be used as redundant I/Os:
   - DI Types 3 and 4
   - DQ Types 10 and 11
   - AI Types 15 and 16
   - AQ Types 20 and 21
5. Fill out the right field in the bottom half of the screen in each form by making the following entries:

   Redundant DIs
   - An L-DI
   - An L-DQ

   } for DIs **with** error locating facility

   - Discrepancy times for the individual DI bits (confirm or change)

Redundant DQs

- An L-DI } for DQs **with** error locating facility
- An L-DQ }
- A Readback DI
- Type specification for the R-DI (one-sided in subunit A or B, switched in the P or O I/O area)

Redundant AIs

- The absolute discrepancy value
- The relative discrepancy value
- The preferred discrepancy value for an AI
- The lower limit of the analog value
- The upper limit of the analog value
- The discrepancy time (confirm or change)

Redundant AQs

- An L-DQ for AQs with error locating facility
- An R-AI
- The absolute discrepancy value
- The discrepancy time (confirm or change)
- The number of updates within the discrepancy time

6. Press <RETURN> after completing the configuration of the I/O.

Putting Redundant I/Os into Operation

The programmer is connected to subunit A or B (power ON). The mode selector switches on both CPUs are at "STOP".

1. Transfer your configuring data to the PLC as described in the section on one-sided I/Os (COM 155H's basic "System Handling" form).
2. Set subunit A to RUN.
3. Load the error DB from the PLC into the programmer and read any entries you find (COM 155H's basic "Diagnostics" form, "H ERRORS" submenu). Rectify any errors you find with the help of your manual.
4. Switch subunit A back to STOP.
5. Repeat the above for subunit B.
6. Now configure the DB/DX numbers for your STEP 5 program in the "TRAFDAT" form.
7. When all errors have been rectified, load your STEP 5 program into the S5-155H.
8. When the complete DX 1 configuring data block has been loaded into the PLC, execute a cold restart on both subunits. Subunits A and B then enter the RUN mode, one as the master, the other as standby (the master's RUN LED will show a steady light, the standby's RUN LED will flash).

Note:
To document your configuring data on paper, have COM 155H print an overview (do so by invoking the "Print" menu in the "System Handling" form).

6.6 S5-155H Responses to Faults/Errors

Sample Program

The example below illustrates how the S5-155H responds to a fault.

FB 37 scans the newest error block for error code YY. If this error/fault is present, the controller can respond accordingly.

STL		Explanation
OB 37		
SEGMENT 1		
	:JU FB37	
NAME	:ERR–EVAL	
	:BE	

STL		Explanation
FB 37		
SEGMENT 1 0000		
NAME	:ERR–EVAL	
0005	:MBA	Address of error DB
0006	:LRW +1	Pointer to current error block
0008	:+D	
0009	:MAB	
000A	:LRW +0	Load –
000C	:L KB 255	error code
000D	:AW	
000E	:L KB YY	Error code YY
000F	:!=F	?
0010	:JC= M001	
0011	:JU =BE	
0012 M001	:	Program the response
0013	:	to error code YY here
0014	:	
0015 BE	:	
0016	:BE	

Time Characteristics of the S5-155H

7

The S5-155H programmable controller's dynamic response differs in a number of ways from that of the S5-155U. This applies to

- Statement execution times
- System program runtimes
- Restart time
- Standby activation
- Function blocks
- On-line programmer functions

7.1 Statement Execution Times for the S5-155H

General Remarks

The majority of STEP 5 statements take the same amount of time to execute in the S5-155H as they do in the S5-155U. The STEP 5 statements which require master-standby synchronization, however, are an exception.

- Direct I/O access operations to a
  - one-sided/two-channel (1-out-of-2) DI — approx. 300 µs
  - three-channel (1-out-of-3) DI — approx. 400 µs
  - one-sided/two-channel (1-out-of-2) DQ — approx. 200 µs
- Direct access to switched I/Os

Statement Execution Times for Access to Switched I/Os

Note:

These execution times depend on

- the length of the cables and
- the modules used (acknowledgement time).

7.2 System Program Runtimes

Increased S5-155H Runtimes

The 155H system program's runtime is longer by the value DT than the 155U system runtime; DT is composed of the following:

- **Self-test slice: T1**

  The self-test slice time can be configured with COM H in increments of 2 ms.

- **I/O test and process image update: T2**

  The purpose of this test is to unify and monitor redundant I/Os, and is executed for those modules only. The increase in the scan time for updating the process image of redundant I/Os is as follows:

– Per switched digital input byte:	25 µs
– Per one-sided digital input byte:	approx. 40 µs
– Per two-channel digital input byte:	approx. 60 µs
– Per three-channel digital input byte:	approx. 120 µs
– Per switched digital output byte:	5 µs
– Per one-sided digital output byte:	approx. 20 µs
– Per two-channel digital output byte:	approx. 40 µs
– Per interprocessor communication input flag:	25 µs
– Per interprocessor communication output flag:	5 µs
– Basic load	2 ms

- **155H system program: T3**

  An almost constant time increase of approximately 5 ms per cycle is needed for bus management, link requests, EU failures and the like.

  In a best-case situation, the scan time increase as compared with the S5-155U is thus

  DT = T1 + T2 + T3

S5-155H Restart Time

The S5-155H restart routine includes a full-scale self-test, thus significantly increasing the S5-155H's restart time as compared with the S5-155U.

The restart time (TA) is as follows:

- For CPUs with 640 Kbytes approx. 8 s
- For CPUs with 1664 Kbytes approx. 20 s

7.3 Dynamic Response on Standby Activation

"Activation" Time

During standby activation, the self-test is inhibited so as not to load the cycle even more. The standby controller is activated in two phases:

- **Phase 1**

Updating of the user program and the constants (see Section 3.3).

During this phase, the self-test is inhibited so as not to load the cycle even more.

Time needed for phase 1:

- With CPU 948R I: 180 PLC cycles
- With CPU 948R II: 436 PLC cycles

- **Phase 2**

Updating of the dynamic data

The amount of time this phase adds to the cycle (on a one-shot basis only, however) depends on several factors:

- Transfer time for flags, counters, timers and RS data
  This is a constant value: T7 = approx. 25 ms
- Transfer time for configured data blocks
  This is an approximate value: T8 = approx. 4 µs / word

The one-shot increase in the scan time caused by standby activation procedures is:

TK = T7 + T8

7.4 Interrupt Response Time

The minimum interrupt response time allowed by the 155H system program is determined by

- the self-test (**Ts**)
- the I/O test (**Te, Ta** and **Tg**).

The time value is calculated from the maximum value for the following four formulas:

1. Ts = 2 ms
2. Te = (n * T-DIr) + (m * T-DIe) + (l * T-DI3) + 1.0 ms
3. Ta = (p * T-DQr) + (q * T-DQe) + 0.5 ms
4. Tg = r * T-DIg

where l, n, m, p, q, r = number of bytes in the process image

T-DI3	=	120	µs	Time for three-channel DIs
T-DIr	=	60	µs	Time for redundant DIs
T-DIe	=	40	µs	Time for single-sided DIs
T-DIg	=	25	µs	Time for switched DIs
T-DQr	=	40	µs	Time for redundant DQs
T-DQe	=	20	µs	Time for single-sided DQs

Normally, the interrupt response time is thus 5 ms, but can increase to as much as 20 ms (128 three-channel DI bytes).

S5-155H Error Diagnostics

8

This chapter discusses all error diagnostics facilities for the S5-155H programmable controller. It describes in detail the structure of the error data block, the block in which the 155H system program enters all errors that are detected, and it also provides a list of error numbers and explains what these codes mean. In addition, it describes in detail the H flag doubleword, error OB 37, and the structure of the H flag word.

8.1 Troubleshooting and Error Handling in the S5-155H

All troubleshooting, error diagnostics and error handling facilities provided for the S5-155U are also available to you in the S5-155H. In addition, the S5-155H provides a number of different options for error identification and error handling. Table 8-1 contains an overview of available facilities.

Automatic error identification and error locating facilities for the S5-155H function better when **no** Block End (BE) operation is programmed in OB 26.

Error Recovery

S5-155H-specific error messages

- The causes of errors reported via COM H must be eliminated in ascending order so that secondary error reports do not make troubleshooting more difficult.
- The elimination of the causes of errors in ascending order is important in an S5-155H system for the following reason:

  A number of errors (PEU, for instance) would cause the S5-155U to stop, but would "only" be reported in an S5-155H system so as not to jeopardize fault tolerance unnecessarily; yet these errors are also critical and must be eliminated as soon as possible.

Table 8-1 Troubleshooting and Error Handling in the S5-155H

	PLC	Brief Description	Error Diag-nostics	Error Hand-ling
Fault LED	155U + 155H	Specific combinations of the STOP/SYSFAULT/INIT/ADF/QVZ and ZYK LEDs on the CPU's frontplate are indicative of specific causes of interruption.	X	
Control bits	155U + 155H	These bits provide information on the current operating status, and report all errors which have occurred up to that point.	X	
ISTACK	155U + 155H	Contains the point of interruption, the bits currently set, the contents of the accumulators, and the reason for the interrupt.	X	
BSTACK	155U + 155H	Lists all blocks which were invoked prior to the STOP but not yet completely processed.	X	
COM 155H "DIAGNOS TICS"	155H only	You can read out all errors which have occurred in the system and user programs up to that point, including point of interruption and time stamp, with the programmer (error DB is output in plaintext).	X	
Error DB	155H only	The 155H system program uses this block to enter all errors detected during the self-test and during program processing, complete with error class, error no. and the date. The block also contains a static error image of all inputs and outputs and all CP/IP interfaces.	X	
Error FBs 19 to 34	155U + 155H	In some cases, the system program invokes the relevant OBs before going into STOP; you can program appropriate responses to errors in these OBs.		X
Error OB 37	155H only	The system program invokes OB 37 each time it detects an error which requires an entry in the error OB; you can program an appropriate response in this OB.		X
H flag doubleword	155H only	This flag doubleword (FD) provides data for the time stamp in the error DB. You can use this flag doubleword to program identifiers helpful in diagnosing errors.	X	
H flag word	155H only	The H flag word's status byte contains important information on the PLC status. You can initiate specific requests in the STEP 5 program via the control byte.	X	

8

8.2 Error Data Block (E-DB)

Entries in the Error DB

You must choose a number between 3 and 255 for this block when you are configuring the system (COM 155H):

```
:
H error DB number (3 to 255): 10
:
```

The 155H system program than generates the error DB automatically in the RESTART routine, normally with a length of 2 Kwords.

The following entries are of particular importance should an error occur:

- **Entry in the error image**

A static error image is stored in the error DB which is organized in such a way that each repairable unit (I/Os, CPs, IPs, IM 314R EU) is assigned its own bit. These bits are arranged in ascending order of addresses or interface numbers. A distinction is also made between repairable units assigned to subunit A and those assigned to subunit B.

All bits in the image are initially set to "0". When the system program detects an error, the appropriate bit is set to "1".

You will find an example under "DW 6 to DW 279: Static Error Image".

- **Entry in the status word**

Each error is also assigned to a specific error group ("Parallel link errors", "I/O errors", and so on). One bit is reserved in the status word for each group. Each of these group error bits remains at "1" as long as the error image shows at least one error belonging to that group. The error groups assigned bits in the status word are listed under "Status Word (DW 3)".

- **Entry in the error record**

Each error detected by the system program is entered in a so-called error record. An error record comprises seven data words. The exact format of an error record is described in detail under "Structure of an Error Record".

As a rule, every error detected is entered only **once** in the data block.

IMPORTANT

Each time an error is entered in the error DB, the 155H system program invokes error OB 37; you can evaluate the error and program an appropriate response in this OB.

Structure of the Error DB

	High 15 … 8	7 … Low 0		
DW 0	Error counter			
DW 1	Write pointer			
DW 2	Unassigned			
DW 3	Status word			
DW 4	Address of the 1st error record (400)			
DW 5	Reserved			
DW 6 ..13	Static error image	DI	0..127	Subunit A
DW 14 ..21	Static error image	DI/AI	128..255	Subunit A
DW 22 ..29	Static error image	DI	0..127	Subunit B
DW 30 ..37	Static error image	DI/AI	128..255	Subunit B
DW 38 ..45	Static error image	DI	0..127	Switched
DW 46 ..53	Static error image	DI/AI	128..255	Switched
DW 54 ..69	Error image of the O area	I	0..255	Switched
DW 70 ..85	Error image of the IPC flags	I	0..255	Switched
DW 86 ..93	Static error image	DQ	0..127	Subunit A
DW 94 ..101	Static error image	DQ/AQ	128..255	Subunit A
DW 102 ..109	Static error image	DQ	0..127	Subunit B
DW 110 ..117	Static error image	DQ/AQ	128..255	Subunit B
DW 118 ..125	Static error image	DQ	0..127	Switched
DW 126 ..133	Static error image	DQ/AQ	128..255	Switched
DW 134 ..149	Error image of the O area	Q	0..255	Switched
DW 150 ..165	Error image of the IPC flag	Q	0..255	Switched
DW 166 ..181	CP/IP interface error image		0..255	Subunit A
DW 182 ..197	CP/IP interface error image		0..255	Subunit B
DW 198 ..213	CP/IP interface error image		0..255	Switched
DW 214	Static error image of EU (IM 314R) Subunit A			
DW 215	Static error image of EU (IM 314R) Subunit B			
DW 216 ..231	Static error image	DI/AI	0..255	Subunit A Q
DW 232 ..247	Static error image	DI/AI	0..255	Subunit B Q
DW 248 ..263	Static error image	DQ/AQ	0..255	Subunit A Q
DW 264 ..279	Static error image	DQ/AQ	0..255	Subunit B Q
DW 280 ..399	Reserved			
DW 400	E. location / E. class	Error code		
	Additional info 1			
	Additional info 2			
	Additonal info 3			
	Prog. number	Ascending number		
	Time stamp, second (BCD)	Time stamp, minute (BCD)		
	Time stamp, hour (BCD)	Time stamp, day (BCD)		
	Time stamp, month (BCD)	Time stamp, year (BCD)		
DW 408	2nd error record			
	and additional error records			

1st error record

8

Data Word DW 0: 'Error Counter'

All errors are tallied in this counter. The counter stops when it reaches 32767. It is reset during a cold restart and on depassivation.

Data Word DW 1: 'Write Pointer'

The write pointer always points to the beginning (the first data word) of the current error record. The current error record is the record in which the most recent error was entered. The write pointer is '0' as long as the error DB contains no errors.

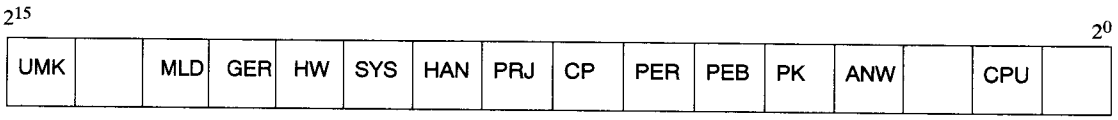
Data Word DW 2: Unassigned

This data word is available to the user.

Data Word DW 3: 'Status Word'

H system errors can be divided into specific groups. Each bit in the status word is reserved for one of these groups, and stays set to '1' as long as the error image shows at least one error for that group.

Data word DW 3 in the error DB is reserved as status word; this word has the following format:



2^0	:		Unassigned
2^1	:	**CPU**	Group error bit for CPU errors
2^2	:		Unassigned
2^3	:	**ANW**	Group error bit for user memory errors (memory card)
2^4	:	**PK**	Group error bit for parallel link errors (IM 304 - IM 324R)
2^5	:	**PEB**	Group error bit for I/O bus errors (IM 304 - IM 314R)
2^6	:	**PER**	Group error bit for I/O errors
2^7	:	**CP**	Group error bit for CP/IP errors (page area)
2^8	:	**PRJ**	Group error bit for configuring errors
2^9	:	**HAN**	Group error bit for data handling errors
2^{10}	:	**SYS**	Group error bit for system errors
2^{11}	:	**HW**	Group error bit for hardware errors (defective CPU, for example)
2^{12}	:	**GER**	Group error bit for device errors (EU power OFF, for example)
2^{13}	:	**MLD**	Message
2^{14}	:		Unassigned
2^{15}	:	**UMK**	'Error DB full' identifier

Bit 15 : 'Error DB full' identifier

This bit is set when all of the data block's error records are full and subsequent entries start again with the first error record (ring method).

DW 4: Address of the 1st error record

DW 4 contains the start address of the first error record in the error DB: Data word number "400".

DW 5: Reserved

DW 6 to DW 279: Static error image

The image in these data words shows which I/Os or CP/IP interfaces are defective.

Example: Static error image of the digital outputs (DQs) assigned to subunit B.

	2^{15}															2^{0}
DW 102	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DW 103	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DW 109	127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112

Digital outputs DQ 30 and DQ 22 in subunit B are bad when bits 14 and 16 in data word DW 103 are "1".

If more than one timeout (QVZ) occurs during updating of the process image (or of the interprocessor communication flags), the byte number with the lowest address is entered **immediately** ; the remaining byte numbers are entered within the next 32 PLC cycles.

Beginning DW 400: Error records

The error records begin in data word DW 400. Each record comprises eight data words.

An error record is filled out for each new error the system program detects during the cycle. When all error records are full, the next error is entered in the first error record (which begins with DW 400) and the 'Error DB full' identifier (bit 15 of the status word) is set to "1".

If several errors are detected during the transfer of the process image to switched I/Os, only one error record entry is made. The other bad addresses are entered in the error image within the next 32 PLC cycles.

Structure of an Error Record

15	8	7	0
E. location	E. class	Error number	
Additional info 1			
Additional info 2			
Additional info 3			
Prog. number		Ascending number	
Time stamp, second (BCD)		Time stamp, minute (BCD)	
Time stamp, hour (BCD)		Time stamp, day (BCD)	
Time stamp, month (BCD)		Time stamp, year (BCD)	

Error Location and Error Class

7	6	5	4	3	2	1	0
B	A	RES	MA		PAS	WST	HST

Bits 0 to 3 contain the error response (default is the standard response):

Bit 0 : **HST** Hard STOP (CPU error, for example)

Bit 1 : **WST** Soft STOP (signature error in user program, etc., for example)

Bit 2 : **PAS** Passivation (on timeout, discrepancy error, etc., for example)

Bit 3 : Unassigned

Bits 0 to 3=0: **MESSAGE**

7	6	5	4	3	2	1	0
B	A	RES	MA		PAS	WST	HST

Bits 4 to 7 describe the error location:

Bit 4 : **MA** Error occurred in master controller

Bit 5 : **RES** Error occurred in standby controller

Bit 6 : **A** Error occurred in subunit A

Bit 7 : **B** Error occurred in subunit B

If, for example, bits 4 and 7 are "1", it is apparent that the error occurred in subunit B, which is currently controlling the process as master.

This produces the following possible combinations:

Bit 6/7 and bit 4/5	= 1: Error occurred in one of the subunits
Bits 6 and 7	= 1: Error occurred in both subunits
Bit 5	= 1: Error occurred in the IM304/IM324R parallel link
Bit 4	= 1: Error occurred in the switched I/Os
Bits 4 and 5	= 1: Comparison error (the Error Search Mode is entered)

8

Error Number

The error number is a consecutive number between 1 and 255; each error number is assigned to a plaintext.

Example: CPU fault, DI module fault, CP/IP fault, etc. (refer to the table "Structure of an Error Record" on the previous page).

Error Location (Additional Information)

The error location informs the user where a specific error has occurred. The information is contained in up to three data words.

Example: Module address, interface no. etc.
(refer to the table "Structure of an Error Record")

Prog. Number and Consecutive Number for the Service Specialist

The number of the system program OB to be executed when an error occurs and a consecutive number (= OB subtask) are entered here.

Time Stamp

The current date and time (from the CPU's system data area) are entered here in the event of an error.

Second (tens)	Second (units)	Minute (tens)	Minute (units)
Hour (tens)	Hour (units)	Day (tens)	Day (units)
Month (tens)	Month (units)	Year (tens)	Year (units)

You can also use the first two data words to store the contents of a specific flag doubleword whose number you specify with COM 155H. Also see "H Flag Doubleword" in Section 8.3.

Note

When several timeouts occur in the switched I/Os during updating of those I/Os (or of the IPC flags), only **one**, i.e., the one with the lowest byte number, is entered in the error record.

The other byte numbers are entered in the static error image.

Table 8-2 Error List

Error No.	Description	Cause of Error	Remedy
2	CPU 948R failure (MPU)	Hardware fault on CPU or CC power supply unit's load voltage monitor not disabled	Check the load voltage or replace the CPU 948R or phone for maintenance.
3	Message from CPU 948R	System error	
4	Clock error (CPU 948R) or battery failure		
7	System error		Phone for maintenance.
8	BASP error		Replace the CPU 948R.
10	I/Os not connected or not ready	I/Os not ready. An EU or I/O bus (721 I/O cable) has failed. If a switched EU has failed, consult the static error image.	
11	Standby failed to activate successfully	Standby stopped again while being activated. The preceding error usually indicates the reason for this error.	Examine the reason why the standby went into STOP by viewing the contents of the ISTACK and the error DB.
13	DX 1 configuring block invalid		Generate or modify DX 1 only via COM 155H.
14	DB 1 invalid. **info 1:** DB 1 ID (ref.) **info 2:** DB 1 ID (act.)		Generate or correct DB 1 via screen form.
15	Master stopped during standby activation	When the master stops during activation, the standby also stops, as the latter has not yet received the current data (flags, for instance).	
16	"Standby STOP" bit set in H flag word	User (program) has set the H flag bit for "Standby has stopped".	
17	I/Os ready again	This message enables you to read or log the time of a repair or to initiate depassivation.	

Error No.	Description	Cause of Error	Remedy
18	One master stopped because partner also master.	Parallel link error	
19	Old standby performs warm restart as master.	This subunit stopped as standby; other subunit was master on last system STOP. The ex-master has the current process data, but is still at STOP.	a) Always use "Cold restart without memory". b) If this is not possible and standby activation is not required, scan OB 37 for this error no. and set the subunit to STOP.
20	Memory module error. **info 1:** Address (page) **info 2:** Address (low)	Memory defective.	Replace CPU.
21	RAM comparison error. **info 1:** Block type/block no. **info 2:** Relative address	A user RAM location does not match in the two subunits, which is not permissible in a redundant system. Assuming there has been no hardware fault, the reason for the RAM comparison error is always one of the following user errors: a) No interrupt DB/DX specified with COM H. b) Illegal access operation to a bit or a word in the system data area (TBRS n.m., LRS n) executed (see list in Section 2.4).	The error can usually be rectified if the specified DB / DX is entered in the list of interrupt DBs/DXs with COM H (see COM H User's Guide).
22	Checksum error in block. **info 1:** Sum (ref.) **info 2:** Sum (act.) **info 3:** Block type/block no.	Contents of the STEP 5 logic block (OB, PB, SB, FB, FX) illegally modified, via OUTPUT ADDR or via the user program.	STEP 5 logic blocks may be modified by the programmer (output block) or OB 124/125 to ensure that the block checksum is correctly entered in DB 0.
23	Error search without result.	Error search found no hardware errors.	Evaluate error code 21 or 24 in the error DB for additional details.

Error No.	Description	Cause of Error	Remedy
24	PIQ comparison error. **info 1:** Address (page) **info 2:** Address (low)	The PIQ is not the same in both subunits. Possible reasons: - Parallel link error - User programming error; for example, illegal access to system RAM in order to change the PIQ.	
25	Operating system EPROM failure.	Bad operating system EPROM.	Replace the CPU 948R.
27	Memory configuration incompatibility.	In a redundant system, the subunits may not have different memory configurations.	Check the SPAUS to see where the difference in the memory configuration lies and upgrade the standby accordingly. After executing an overall reset on the standby, it can be reactivated (see error no. 21 for further details).
28	System program incompatibility.	In a redundant system, the subunits may not have EPROMs with different contents.	Activating the standby when the subunits are using two different operating system versions is permissible only when the standby is the one with the newer version. Prior to activation, the COM H "UPGRADE" parameter must be set to "CPU 948R". The standby can then be activated, and takes over as master. The other subunit stops and must then be upgraded.
29	Different user program code.	The subunits cannot be operated with different user memory cards (the block sequence must also be the same!).	Generate two identical memory cards and activate the first using COM 155H's "UPGRADE", "MEMCARD" function (see Chapter 9, "Upgrading").
30	Standby-master switchover due to master failure **info 1:** Instruction code **info 2:** Absolute SAC (high) **info 3:** Absolute SAC (low)	The master failed during user program processing.	

Error No.	Description	Cause of Error	Remedy
31	Standby failure. **info 1:** Instruction code **info 2:** Absolute SAC (high) **info 3:** Absolute SAC (low)	Standby controller failure during user program processing.	
32	Synchronization error (master-standby) in the user program. **info 1:** Instruction code **info 2:** Absolute SAC (high) **info 3:** Absolute SAC (low)	A subunit waited at this synchronization point until the waiting time elapsed. One reason for this error might be differences in the program scan in master and standby caused by discrepancies in DB contents.	All DBs (DXs) whose contents have been changed in an interrupt handling routine must be reconfigured as interrupt DBs/DXs.
33	Parallel link error (IM 304 / IM 324R)	IM 304, IM 324R or parallel link cable failed.	
34	Synchronization error (master-standby time) in the operating system	For maintenance purposes.	Print out the error DB, the ISTACK and the BSTACK and phone for maintenance.
35	Standby-master switchover due to master failure	Master failed while the operating system was processing.	
36	Standby failure	Standby failed while the operating system was processing.	
37	Standby-master switchover due to I/O error. **info 1:** QVZ address (page) **info 2:** QVZ address (low)	A standby-master switchover is carried out automatically in the event of a timeout (QVZ) on a switched I/O because the operating system assumes that an I/O bus has failed. The subsequent error message(s) and the static error image show what it was that failed (an EU, for instance). Any additional timeouts occurring prior to the next passivation do not cause another transfer.	Rectify the I/O error. Fault-tolerance is restricted because the master's bus failure prevents a standby-master switchover.
38	Synchronization error (master-standby location) in the user program. **info 1:** Instruction code **info 2:** Abs. address (high) **info 3:** Abs. address (low)	A subunit has recognized that its peer is waiting at another synchronization point. See error no. 32 for the cause of error and suitable recovery measures.	

Error No.	Description	Cause of Error	Remedy
39	Synchronization error (master-standby location) in the operating system.	For maintenance purposes	Print out the error DB, ISTACK and BSTACK and phone for maintenance.
40	I/O bus error (IM 304 / IM 314R) **info 1:** EU number	EU power supply or IM 314R / IM 304 module or I/O bus cable failure.	Check to see if more than one error with no. 40 was entered in the same cycle. Following repairs, H flag bit DPA (depassivation) must be set.
50	Timeout (QVZ) on input module **info 1:** QVZ address **info 2:** (Instruction code)	Module fault or EU failure. Is sometimes reported together with error no. 40 or 10, which might be the cause of error.	
51	Timeout (QVZ) on output module **info 1:** QVZ address **info 2:** (Instruction code)	See error no. 50.	
52	Timeout (QVZ) when accessing IB 0 (process interrupt active) **info 1:** QVZ address **info 2:** Instruction code	See error no. 50.	
53	Timeout (QVZ) on input communication flag **info 1:** QVZ address **info 2:** Instruction code	Incorrect jumper setting on CPs/IPs or EU failure or module fault. See error no. 50.	
54	Timeout (QVZ) on output communication flag **info 1:** QVZ address **info 2:** Instruction code	See error no. 53.	
55	Configured DI byte not found (timeout) **info 1:** DI address	Configuring error or incorrect jumper setting, module fault or EU failure.	
56	Configured DQ byte not found (timeout) **info 1:** DQ address	See error no. 55.	

Error No.	Description	Cause of Error	Remedy
57	Configured AI byte not found (timeout) **info 1:** AI address	See error no. 55.	
58	Configured AQ byte not found (timeout). **info 1:** AQ address	See error no. 55.	
59	DQ group passivation (L-DQ or DQ error) **info 1:** DQ address	When a locating digital output (L-DQ) fails, the DQ modules assigned to it in the same subunit are passivated.	First, check to see if error code 71, 73 or 74 has been entered in the error DB.
60	DI error (stuck at "0" or "1") **info 1:** Bit no. / byte no.	The DIs still do not match although the discrepancy time has elapsed. The error info includes the subunit containing the bad module.	
61	DQ error (stuck at "0" or "1") **info 1:** Bit no. / byte no. **info 2:** 0: stuck at 0 1: stuck at 1	The readback digital input (R-DI) did not read back the anticipated value although the readback delay had expired.	If the same error is reported at the same time (immediately before or after) for the other controller, it is indicative of an external error. The R-DI is subsequently passivated.
62	AI error (wirebreak) **info 1:** AI address **info 2:** Channel number	In addition to an error bit (wirebreak, for example), this error is also reported by function block "H-RLG:AE".	
63	AI error (range violation) **info 1:** AI address **info 2:** Channel number	See error no. 62.	
65	AI error (discrepancy error) **info 1:** AI address **info 2:** Channel number	See error no. 62.	
66	AQ error **info 1:** Byte no.		

Error No.	Description	Cause of Error	Remedy
68	DI error on 3-channel DI. **info 1:** Bit no. / byte no. **info 2:** 0: stuck at 0 1: stuck at 1	The DIs still do not match although the discrepancy time has elapsed. The error info specifies the side on which the problem was detected. The problem may be a wirebreak, a DI module fault or a sensor fault.	
69	Discrepancy timeout for DI bit – no signal changed. **info 1:** Bit no. / byte no.	A discrepancy error was detected on a 2-channel DI without error locating facility; the error could not yet be located. Error locating begins with the next edge change at this input.	
70	Error in locating circuit for DI. **info 1:** Bit no. / byte no. of L-DI **info 2:** Bit no. / byte no. of L-DQ **info 3:** Stuck at 0 or stuck at 1 error	This error no. is indicative of a wiring or module fault on the L-DI or L-DQ module. The error info includes the byte number and the bit number of the L-DQ and the L-DI, as the fault could be on **either**.	
71	Error in locating circuit for DQ. **info 1:** Bit no. / byte no. of L-DI **info 2:** Bit no. / byte no. of L-DQ **info 3:** Stuck at 0 or stuck at 1 error	See error no. 70. Error in the error locating facility for redundant DQs.	
72	Timeout (QVZ) on "error locating DI". **info 1:** Byte number of L-DI	L-DI module defective or failure of the EU containing the L-DI module.	
73	Timeout (QVZ) on "error locating DQ". **info 1:** Byte no. of L-DQ	L-DQ module defective or failure of the EU containing the L-DQ module.	
74	Stuck at 0 error of an L-DQ for redundant DQs. **info 1:** Bit no. / byte no. of the L-DQ	L-DQ bit defective or wiring fault.	

Error No.	Description	Cause of Error	Remedy
75	Readback error. **info 1:** Byte no. of R-DI **info 2:** Byte no. of associated DQ	If error 61 is reported at the same time (just before), it has priority. Error no. 75 only indicates that the R-DI is no longer in use. Otherwise, this error no. means only that there is a problem with the R-DI or the wiring.	
76	Timeout (QVZ) on readback DI. **info 1:** Byte no. of R-DI **info 2:** Byte no. of associated DQ	R-DI module defective or failure of the EU containing this R-DI.	
77	Stuck at 1 – redundant readback DI. **info 1:** Bit no. / byte no. **info 2:** Byte no. of the associated redundant DQ	The readback input detected signal state "1" although the redundant output is supposed to output "0".	
78	Timeout (QVZ) in redundant readback DI. **info 1:** Byte no. causing the timeout **info 2:** Byte no. of the associated redundant DQ	Module or EU failure.	
79	Stuck at 0 in redundant readback DI or in redundant DQ. **info 1:** Bit no. / byte no. **info 2:** Byte no. of the associated redundant DQ	Output or readback input defective.	
80	CP/IP interface error. **info 1:** Interface no. (page number)	CP/IP was not configured or CP/IP acknowledges with more than one interface number.	
81	CP/IP does not acknowledge (timeout). **info 1:** Interface no. (page number)	CP/IP fault or failure of the EU containing the CP/IP.	
85	Input communication flag not available. **info 1:** Address (F page)	Module fault, PLC failure or configuring error. The interprocessor communication input flag configured in DB 1 does not exist. In contrast to the 155U, the 155H allows a cold restart/warm restart without IPC flag.	

Error No.	Description	Cause of Error	Remedy
86	Output communication flag not available. **info 1:** Address (F page)	See error no. 85.	
90	Cannot locate configured DB/DX. **info 1:** High: 0 = Cyc DB 1 = Cyc DX 2 = Interr. DB 3 = Interr. DX Low: DB/DX number	Configuring error. Note: This error increases the length of the activation cycle.	
91	I/O communication flag ID set more than once in DB 1. **info 1:** Illegal identifier	DB 1 not generated via screen form.	Generate DB 1 via screen form.
92	Area ID (I/O comm. flag) missing in DB 1.	See error no. 91.	
93	I/O communication flag byte no. > 255 (DB 1) **info 1:** IPC input/output flag ID **info 2:** Illegal byte number	See error no. 91.	
94	> 255 I/O communication flag bytes (DB 1). **info 1:** IPC input/output flag ID	See error no. 91.	
95	DI module not configured. **info 1:** DI address	Some of the modules being used have not been configured, or a module was incorrectly addressed or acknowledges for more than one address.	
96	DQ module not configured. **info 1:** DQ address	See error no. 95.	
97	AI module not configured. **info 1:** AI address	See error no. 95.	
98	AQ module not configured. **info 1:** AQ address	See error no. 95.	

Error No.	Description	Cause of Error	Remedy
99	DB/DX not configured in TRAFDAT list. **info 1:** Low: DB/DX number High: 0 = DB, 1 = DX	This DB/DX is modified by the user program.	Check to see whether DB/DX is modified in the cyclic program or in an interrupt service routine, and configure the DB/DX in the appropriate list.
101	Prog. attempted illegal I/O access. **info 1:** Instruction code **info 2:** Abs. SAC (high) **info 3:** Abs. SAC (low)	This programming error is detected in redundant systems only, and means that one of the following was not observed: a) Block transfers to I/Os must not involve the address area for redundant/one-sided I/Os; refer to the information provided on initializing operating system parameters. b) Read access by byte to a redundant AI is not permitted; redundant AIs must be accessed by word in order to enable result-matching.	
105	GDB error in the user program.	The interval required between block deletion and block generation was not observed.	
110	Internal CPU 948R communication error.	Interrupts disabled too long.	
121	CPU 948R data bus error.	CPU 948R hardware fault.	Replace CPU 948R.
122	CPU 948R addr. bus error.	CPU 948R hardware fault.	Replace CPU 948R.
123	CPU 948R EPROM error.	CPU 948R hardware fault.	Replace CPU 948R.
124	CPU 948R RAM error.	CPU 948R hardware fault.	Replace CPU 948R.
125	CPU 948R CPU error.	CPU 948R hardware fault.	Replace CPU 948R.
132	Discrepancy error in 3rd DI channel. **info 1:** Byte no. for discrepancy. **info 2:** No. of ass. byte in A and B **info 3:** DL: Permanent 1/0 DR: Bit no.	DIs still do not match although discrepancy time has elapsed. Error info includes the subunit containing the bad module.	
135	Timeout (QVZ) on 3rd AI channel **info 1:** Byte no. reporting the timeout. **info 2:** No. of the associated AI in A and B	Module fault or EU failure. May be reported together with error no. 40 or 10, which could be the cause of the error.	

Error No.	Description	Cause of Error	Remedy
136	Discrepancy error in 3rd AI channel. **info 1:** AI address **info 2:** No. of the associated channel in A and B	The third AI channel differs from the other two channels for a longer period than that defined by the discrepancy time.	
137	Wirebreak in 3rd AI channel. **info 1:** AI address **info 2:** No. of the associated channel in A and B		
138	Overflow in 3rd AI channel. **info 1:** AI address **info 2:** No. of the associated channel in A and B		
139	Range violation in 3rd AI channel. **info 1:** AI address **info 2:** No. of the associated channel in A and B		
140	Timeout (QVZ) in readback analog input. **info 1:** Readback AI address **info 2:** Associated AQ address	Module or EU failure.	
141	Wirebreak in readback analog input. **info 1:** Readback AI address **info 2:** Associated AQ address	Wirebreak.	
142	Discrepancy in readback analog input. **info 1:** Readback AI address **info 2:** Associated AQ address	The second analog output has already been passivated or the PLC is running in Solo mode. If a discrepancy on this analog channel was reported within 5 minutes, it is not the AQ, but rather the AI, that is defective.	

Error No.	Description	Cause of Error	Remedy
143	Discrepancy: AQ value too high. **info 1:** AQ address **info 2:** Actual discrepancy value	The readback analog input reads a higher value than the one to be output to the analog output.	Check/re-measure output value and replace defective module.
144	Discrepancy: AQ value too low. **info 1:** AQ address **info 2:** Actual discrepancy value	The readback analog input reads a lower value than the one to be output to the analog output.	Check/re-measure output value and replace defective module.
145	Timeout (QVZ) in locating DQ.	Module or EU failure.	
150	DB/DX too short or list of dyn. DB/DX unequal. **info 1:** Block length in master controller **info 2:** Block length in standby controller **info 3:** Block number / identifier in master controller	The user program on the standby controller's new MEMCARD differs from the master controller's user program, and this difference is not permitted.	Copy DB/DX block to a new MEMCARD or increase block length.
151	Depassivation due to standby activation.	The master executed automatic depassivation as part of the standby activation phase. This error is entered in the error list only when at least one fault/error is reported in the master controller.	
152	Depassivate without deleting the error records.	H flag control bit X.1 set.	

Explanation of Additional Information

Interface no.: Interface number for CP/IP errors

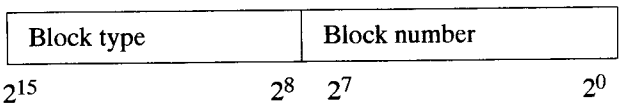
SAC: STEP address counter. The STEP address counter points to the absolute address +1 in the program memory of the last operation executed.

Explanation of Additional Information

Interface no.: Interface number for CP/IP errors

SAC: STEP address counter. The STEP address counter points to the absolute address +1 in the program memory of the last operation executed.

Block type/block number: This block was the last one processed.



Block type			Block number
1	=	DB	0 to 255
2	=	SB	
4	=	PB	
5	=	FX	
8	=	FB	
12	=	DX	
16	=	OB	

IMPORTANT

The error DB is erased **in its entirety**

- when a cold restart is performed on the master controller
- when you request depassivation by setting the appropriate bit in the H flag word.

Notes on Error Diagnostics

Not all error numbers are reserved.

Avoid RAM comparison errors (error search mode) by

- configuring all DBs/DXs which are modified in an interrupt OB (OBs 2 to 18) as **interrupt** DBs/DXs (refer to the COM 155H User's Guide).

Evaluating the Error Data Block

You can evaluate the error DB in various ways.

a) **Evaluating the error DB on the programmer with COM 155H:**

Using the COM 155H "DIAGNOSTICS" function, you can have the errors in the error data block output in plaintext. You can scroll up and down through the error records on the screen.

b) **Evaluating the error DB with STEP 5:**

Since the system program automatically invokes the OB 37 user interface when an entry is made in the error DB, it is easy to evaluate the contents of the error DB in OB 37 as part of the STEP 5 program in that organization block (with the aid of error counters, read and write pointers, status word, and so on) and then to respond suitably to that error (refer to the sample program in Section 6.6).

c) **Evaluating the error DB on the programmer using online functions:**

Using a series of on-line functions, you can read out the error data block as a data field right directly on the programmer.

Error Messages from the CP 523

The example in this section shows you how to use OB 37 to output the operating system error messages via CP 523

Preparing the CP 523

1. Plug the memory card into the programmer.
2. Insert the COM 155H diskette in the disk drive and copy the following blocks from file DB523DST.S5D to the memory card.
   - DB 1 Parameter initialization DB (parameters for page length, etc.; refer to the CP 523 manual).
   - DB 194
   - DB 195
   - DB 196
3. Insert the memory card into the receptacle on the CP 523.
4. Set the required module address on the CP 523 (refer to Chapter 5 in the CP 523 manual).

Note

Data block DB 1 contains the printer configuration. The following parameter default values are already set:

- V.24; 9600 bps; 2 stop bits; 7 data bits; 1 start bit; no parity (corresponds to the programmer defaults).
- Number of lines: 72.
- If you want a different configuration, you must modify the relevant parameters in DB 1.

Outputting Error Messages

Error information is passed to the CP 523 via an "Error message block"; FB 48 (which is part of the COM 155H package and stored as file "S5CR70ST.S5D"), which you must call in OB 37.

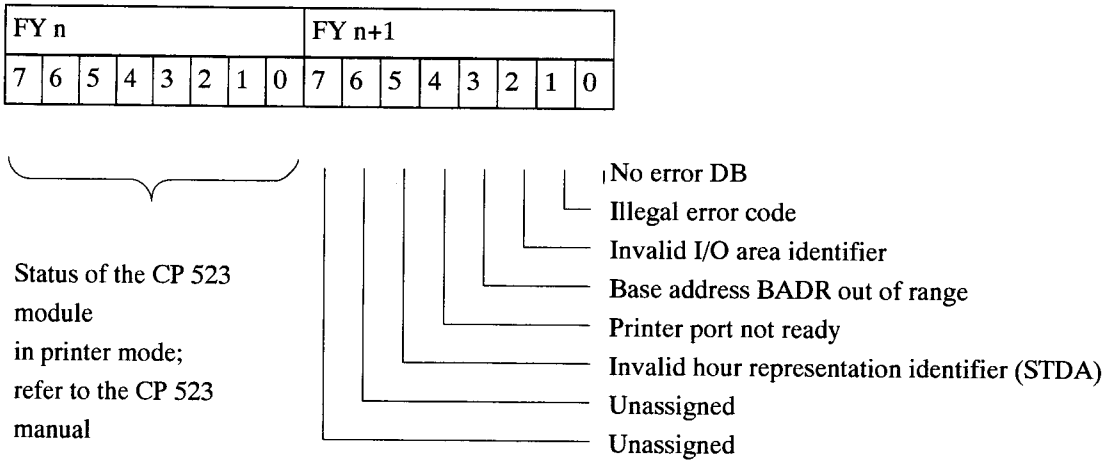
Calling parameters in OB 37:

STL	Explanation
OB 37	
:JU FB 48	Standard error messages CP 523
NAME:CP 523 STF	
BADR:KF +128	
P/Q :KS O	
STDA:KS 24	
CPUZ:KS JJ	
FEWO:FW 195	
:BE	

Name	Parameter-Type	Data Type	Description	Remarks
BADR	D	KF	Module address	P/Q = P : BADR = 128 .. 248 P/Q = Q : BADR = 0 .. 248
P/Q	D	KS	I/O area	P/Q = P: P area P/Q = Q: O area
STDA	D	KS	Hour representation	STDA = 24 : 24-hour representation (German) STDA = 12 : 12-hour representation (US)
CPUZ	D	KS	PLC clock or CP clock as time source	CPUZ = JJ : The PLC clock is used as time source, making the COM H FM time and the printer time the same. Each error reported resets the CP clock. CPUZ = NN : The CP clock is used as the time source. The COM H error message time and the printer output time may differ.
FEWO	Q	FW DW	Value address for error word	All errors detected by FB 48 during parameter initialization and processing

Note:
Even when set to the O address area, the CP can be plugged into an EU 185 which is set for the P address area.

Description of the Error Word (FEWO) in FB 48



Sample printout

```
                        :
10.05.91  11:03:01    SUBUNIT B STANDBY SOFT STOP E-NO.: 020
                      MEMORY MODULE FAULT
                      ADDRESS:  0003  0001  H
10.05.91  11:03:01    SUBUNIT A STANDBY PASSIVATION E-NO.: 061
                      DI ERROR (STUCK AT 0 OR STUCK AT 1)
                      BIT NO./BYTE NO.  005,003
                      STUCK AT 001 ERROR
```

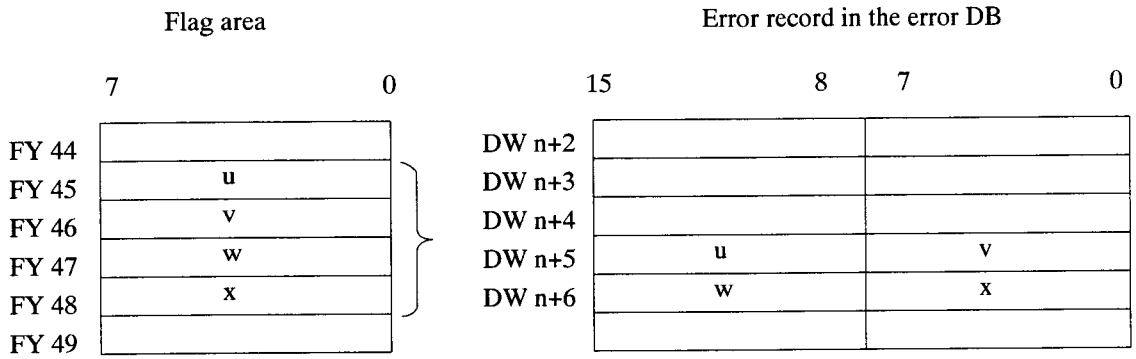
8.3 H Flag Doubleword

Using the H Flag Doubleword

The contents of this flag doubleword, whose number you yourself may choose and stipulate in a COM H run, supplies the time stamp (in the 6th and 7th data word of each error record) for all error messages in the error data block.

You can, for example, use the bits in this doubleword for information useful in error diagnostics (cycle counter or sequencer status).

Example for flag doubleword FD 45:



If, when configuring the system with COM 155H, you do **not** specify a number in the line

```
:
Time stamp flag doubleword (SEC/0..252):SEC
:
```

(see "SEC" parameter) the internal CPU time is automatically stored in the time stamp in the event of an error (provided the real-time clock has been set). This ensures that every error entry is provided with the exact time (second, minute, hour) and date (day, month, year).

8

8.4 Error Organization Block OB 37

Tasks Performed by OB 37

As soon as the 155H system program detects an error (during execution of the self-test, for example) and enters it in the error data block, it invokes organization block OB 37.

OB 37 is the block in which you can program the desired responses to errors; for example to output an error message via the CP 523 and/or set the PLC to STOP.

If more than one new 155H-specific error occurs in the STEP 5 program which makes up OB 37, only the first error is entered as error record.

8.5 The H Flag Word

The H flag word contains important information about the status of your programmable controller (such as "PLC in Error Search Mode"), which you can also evaluate in OB 37.

You may select the number for the H flag word yourself, and specify it during your configuring session with COM H:

```
:
H system flag word      (0..254):    0
:
```

Format of the H Flag Word

The H flag word consists of a status byte and a control byte.

The control information can be set bit by bit in the STEP 5 user program.

The information provided in the status byte is read out in the user program.

Status Byte (High-Order Byte of FY 0)

7	6	5	4	3	2	1	0
AGF	ADL	AMA	ZGA	ANK	MAS	RED	SOL

Bit 0 : **SOL**
"Solo mode"
"1" : PLC is in "Solo mode"
"0" : in all other cases

Bit 1 : **RED**
"Redundant mode"
"1" : PLC is in "Redundant mode"
"0" : in all other cases

Bit 2 : **MAS**
"Master"
"1" : CC is master
"0" : CC is standby
} In redundant mode this bit is always 0

Bit 3 : **ANK**
"Standby activation"
"1" : PLC is in "Standby activation" mode
"0" : in all other cases

Bit 4 : **ZGA**
Central controller is subunit A/B
"1" : CC is subunit A
"0" : CC is subunit B
} In redundant mode this bit is always 0

Bit 5 : **AMA**
Subunit
"1" : Subunit A is master
"0" : Subunit B is master

Bit 6 : **ADL**
Updating in progress
"1" : Standby is being updated
"0" : in all other cases

Cyclic DBs/DXs are transferred.
The system program automatically resets this bit when updating has been completed. The bit can be evaluated only in interrupt OBs.

In order to keep the number of interrupt DBs/DXs to a minimum, this bit can be scanned in interrupt service OBs in order to avoid modifications to the contents of DBs for this brief period of time.

Bit 7 : **AGF**
Note: **No** automatic switchover possible
"1" : PLC fault in switched I/Os
"0" : No faults

(PLC faults are, for example, I/O bus failure, DI 0 failure in one subunit, timeout for switched I/O, etc. An automatic switchover is no longer possible should a secondary fault occur)

Control Byte (Low-Order Byte, e.g. FY 1)

7	6	5	4	3	2	1	0
	RST	AUM	DPA		ARE	DPO	OAT

Bit 0 : **OAT** (No Restart test)
Suppress Restart test
"1" : Restart master without test
"0" : Restart master with test

The system program resets this bit after cold restarts and overall resets.

Bit 1 : **DPO** (Without depassivation)
Depassivate without deleting error records

Important:
This bit **must** be set using signal edge evaluation; that is, it may be set once only, and may not be set again until the system program has reset it. The system program resets the bit after passivation has been revoked. This bit may not be reset in the user program.

Bit 2 : **ARE** (Update standby)
Disable updating of the standby controller
"1" : Updating of the standby controller disabled
"0" : Updating of the standby controller enabled (= default)

Bit 3 : Reserved

Bit 4 : **DPA** (Depassivation)
Revoke passivation
"1" : Revoke passivation
"0" : in all other cases (default)

Important:
This bit **must** be set using signal edge evaluation; that is, it may be set once only, and may not be set again until the system program has reset it. The system program resets the bit after passivation has been revoked. This bit may not be reset in the user program.

Bit 5 : **AUM** (Switchover request)
Switchover requested
"1" : Switchover requested "0" : in all other cases (default)

The system program resets this bit after the switchover has taken place.

Bit 6 : **RST** (Standby STOP)
Set standby controller to STOP
"1" : Set standby controller to STOP
"0" : in all other cases (default)

The system program resets this bit after setting the standby to STOP.

Bit 7 : Unassigned

Dynamic Response to Faults, Repair, Replacement and Upgrading

9

This chapter discusses the S5-155H's dynamic response to faults and to module, expansion unit and cable failures, and recommends repair procedures.

It also explains how to replace the memory card and update the CPU.

9.1 Failure and Repair of the CPU and Parallel Links

Replacing the Central Processing Unit

If you have to replace a CPU 948R because of a defect, you will also have to replace the other CPU 948R if it is not running the same version of the operating system. To replace a CPU 948R without interrupting operation, proceed as described in Section 9.6.

Replacing the Interface Modules (IMs)

Should the parallel link between subunit A and subunit B fail, because of a cable break or a module defect, for example, the master controller continues in Solo mode and the standby controller stops.

Proceed as follows to replace a component:

- First, replace the 721 cable that connects the IM 304 with the IM 324R and start the standby controller.
- If the link fails again, set the standby controller to "STOP". Switch off the power. Replace the standby controller's IM 304 (or IM 324R) interface module. Reconnect the IM 304 and the IM 324R and start the standby controller.
- If the link fails again, proceed exactly as described below, for only then will you be able to process your program in NON-STOP mode.

1. Set the standby controller to "STOP" and switch off the power.
2. Remove the IM 304 or IM 324R from the standby controller and plug in a functional IM 304.
3. Case-basis decision:
   A If the master is equipped with an IM 324R, go to 4.
   B If the master is equipped with an IM 304, it must not be removed; an IM 304 cannot be inserted/removed while the controller is in operation (the master would stop). **A slot (27 to 123) must be available for the new IM 324R.** (After repairs, the IM 304 and the IM 324R will each be in the opposite subunit, but this does not affect the subunit identification). Now go to 5.
4. Connect the 24 V supply and ground to the appropriate terminals on the frontplate of the IM 324R. You can use the 24 V output of your central controller's power supply unit for this purpose. **Now remove this IM 324R.**
5. Connect the 24 V and ground to the appropriate terminals on the frontplate of the IM 324R (you can use the 24 V output of your central controller's power supply unit for this purpose). Plug the IM 324R into the controller that is currently the master.
6. Remove the 24 V source (including ground). The green LED on the IM 324R's frontplate goes on. Reconnect the IM 304 and the IM 324R.
7. Switch the standby controller on and start it up.

After repairs have been completed and the standby controller has been activated, the S5-155H is once again fault tolerant.

Replacing the IM 30x Interface Modules in the CC (Excepting those used for the Parallel Link)

The relevant central controller must always be at STOP and its power switched off.

Replacing the IM 30x – IM 31x Connecting Cable

In one-sided/redundant systems, the expansion unit must be switched off when replacing the 721 cable connecting an IM 30x and an IM 31x.

In switched configurations, only the standby's 721 cable may be replaced when it connects an IM 304 and an IM 314R. Before doing so, always set the standby to STOP.

9.2 Failure and Repair of Expansion Units (EUs)

One-sided EUs

A "cable break" or "EU power failure" is reported in the form of one of more error messages. You can evaluate the timeout information for the relevant I/O modules. The I/O addresses of these modules are passivated, as otherwise there would be a continuous timeout. The S5-155H tolerates the failure of all one-sided EUs; that is, both CCs continue operation even without EUs.

When an expansion unit has been repaired, you must depassivate that unit (by setting the "DPA" bit in the H flag word). The modules plugged into that unit are then once again entered in the process image. Following activation, the S5-155H is once again fault tolerant.

Switched EUs

If an IM 304 (in the master) or IM 314R (in an expansion unit) fails because of a cable break (plug pulled), for instance, the 155H system program switches to the standby. The current master becomes the standby, and reports a fault.

When an EU fails, an "I/O bus fault" is reported (error no. 40); the user can then evaluate the fault information.

The S5-155H tolerates the failure of all switched expansion units; that is, both controllers continue to operate, even without EUs. If one of the two controllers in a switched configuration fails, the other continues in Solo mode.

When a switched EU has been repaired, it is automatically activated in the next PLC cycle. The modules in that EU are updated.

Redundant EUs

When an EU fails, the central controller continues to run. The fault is reported and entered in the error data block.

For information on repairs, see "One-sided EUs".

Following depassivation, the S5-155H is once again fault tolerant. The one-sided I/Os are updated.

9.3 Failure and Repair of I/O Modules

Replacing an I/O Module

When an I/O module fails, the failure is detected in the S5-155H

- through a timeout (QVZ) or
- during the self-test.

The module failure is reported.

You can replace the module during operation if you

1. Use the enable input and
2. Remove the front connector before changing the module.

When the I/O module has been repaired and is back in the subrack, it is not activated until it has been depassivated (by setting the "DPA" bit in the H flag word). The module is once again entered in the process image. When the activation phase has been completed, the S5-155H is once again fault tolerant.

9.4 Failure and Repair of CP/IP Modules

Replacing a CP/IP

In the S5-155H, the failure of a CP/IP module is detected because of a timeout (QVZ). The module failure is reported. "CP/IP does not acknowledge" is entered in the error DB (a timeout (QVZ) has error no. 81).

1. Switch off the power to the subrack (EU or possibly the CC) containing the defective CP/IP module before removing it for repairs or replacing it with another module.
2. Once the CP/IP has been replaced, you must revoke its passivation (by setting the "DPA" bit in the H flag word). You must also invoke the "SYNCHRON" function block (FB 125), which prepares the module for reinclusion.

   The module is then activated. Once the activation phase has been completed, the S5-155H is once again fault tolerant.

Sample Program

```
STL                     Explanation
     :                  Program for CP 1
     :O F   10.0        F 10.0 must be set when CP 1
     :L KT  150.2       is repaired
     :SE T  1           Depassivation takes max. 15 sec.
     :O(
     :L     FY 8        PAFE byte for CP 1
     :L KH  0000
     :><F
     :)
     :O F   10.0        F 10.0 must be set when the
     :R F   10.0        CP 1 is repaired
     :=F    11.0
     :JC    FB 125
NAME :SYNCHRON
SSNR :      KY0.0       Page no. for CP 1
BLGR :      KY0.6
PAFE :      FY 8
     :A F   11.0        Depassivation requested?
     :A T   1           Have 15 seconds elapsed yet?
     :BEC               Yes -> BE, as only one CP
     :                  may be depassivated at a time
     :A F   11.0        Have 15 seconds elapsed?
     :R F   11.0
     :JC    FBxx        Error reported; CP 1 cannot
     :L KB  0           be depassivated
     :T FY  8
```

(Continued on next page)

STL		Explanation
:		
:		**Program for CP 2**
:O F	10.1	F 10.0 must be set when
:L KT	150.2	CP 2 is repaired.
:SE T	1	Depassivation takes max. 15 seconds
:O(		
:L	FY 9	PAFE byte for CP 2
:L KH	0000	
:><F		
:)		
:O F	10.1	F 10.1 must be set when CP 2 is
:R F	10.1	repaired.
:=F	11.1	
:JC	FB 125	
NAME :SYNCHRON		
SSNR:	KY0.2	Page no. for CP 2
BLGR:	KY0.6	
PAFE:	FY 9	
:A T	1	Only one CP may be
:BEC		depassivated at a time
:		
:A F	11.1	
:R F	11.1	
:JC	FBxx	Error reported; CP 2 cannot
:L KB	0	be depassivated
:T FY	9	
:		
:		**Program for CP 3**

9.5 Standby-Master Switchover

Switchover Criteria

During operation, a standby-master switchover results in the current master becoming the standby and the current standby becoming the master. Such a switchover takes place when

a) the master fails (BASP, NAU or mode selector set to 'STOP')

b) the first error search (see Section 2.5) on the standby was unsuccessful

c) an IM 314R fails

d) an I/O bus to a switched EU fails

e) a switched I/O module fails

f) the user requests a switchover at the software level (by setting a bit in the H flag word or via COM 155H).

In situation a), the standby controller enters the Solo mode. In case b), the **new** standby enters the Error Search mode while the "old" standby continues in Solo mode. In cases c) to f), the **new** standby controller does not stop, but continues on as standby.

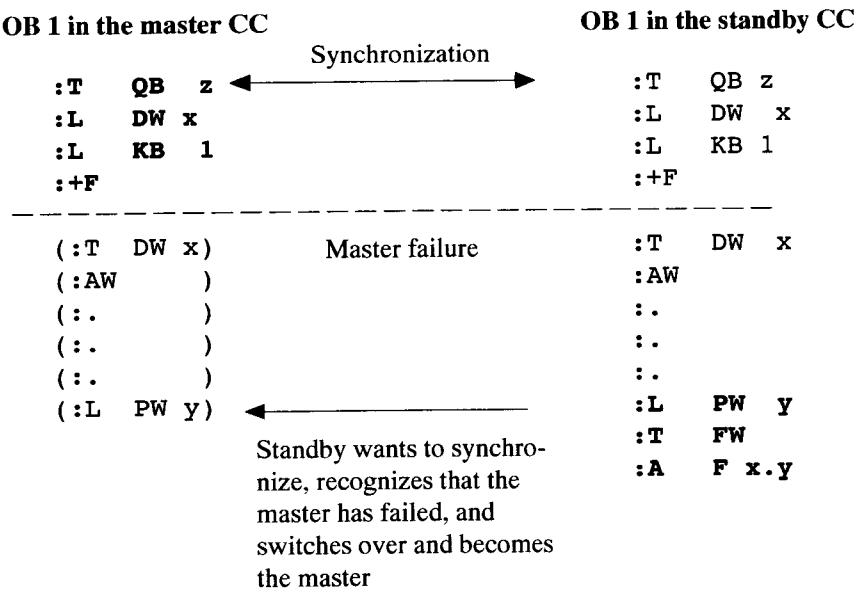
Standby-Master Switchover when the Master Fails

Via the IM 324R parallel link module, the standby controller checks at every synchronization point to see if the master is ready. If the standby ascertains that the master has failed, it takes the following actions:

- The I/O buses of all IM 314Rs are switched.
- The two-channel I/Os are switched to one-channel operation.
- The "switchover mode" is entered; that is, the subunits are no longer synchronized.
- If the synchronization point is a direct I/O operation to a switched I/O, the operation is repeated.

Example

Standby-Master Switchover



One-Sided I/Os in the Failed Subunit

The one-sided I/Os assigned to the failed subunit are treated as follows:

- The process output image (PIQ) is set to "0".
- The process input image (PII) is set to "0".
- A timeout (QVZ) is reported in the event of a direct access operation to these I/Os or this PII/PIQ.

Note

One-sided I/Os should be used only for **independent subprocesses**, which are completely passivated in the event of a PLC failure. The software for controlling these processes should be stored in separate blocks and called only **conditionally**; that is, only when the respective subunit is operating.

Bumpless Switchover

Event-driven synchronization ensures bumpless standby-master switchover whenever necessary; such a switchover has no effect whatsoever on process output signals, and there is no loss of information during communication with the CPs/IPs.

Switchover Time

The switchover time is the once-only increase in scan time caused by a standby-master switchover. From the instant at which the fault/error (synchronization error, for example) is detected, the switchover itself takes no more than 30 ms (full configuration).

The switchover takes up to 3 ms in the event of a CC power supply failure or total CPU failure, or when a request for a switchover is made via H flag (full configuration).

9.6 Replacing the Memory Card During Operation

Conditions and Upgrading Sequence

Modifications to the user program in the flash EPROM memory card must be made known to the operating system via the "software modification" parameter. When the standby is activated, this parameter prevents a comparison between the 32-bit checksums of the controllers' memory cards, and the master CPU's user program is not transferred to the standby CPU.

All data blocks in the standby must come from the memory card. The new data blocks may be longer, but not shorter than those in the master.

The contents of the data blocks in the COM H list, timers, counters, flags and system data are transferred at one synchronization point. Upon completion of the activation phase, a standby-master switchover is executed and and CPU which is now the standby stops. The "software modification" parameter is reset.

Operator Input Sequence

The following entries are required:

1. Using COM 155H, set the "PLC memory card upgrade function" parameter. This effects deactivation of Restart functions "DB 0 comparison" and "Block transfer".
2. Set the standby controller to STOP.
3. Replace the standby CPU 948R's memory card.
4. Execute an overall reset on the standby controller.
5. Execute a cold restart by setting the standby controller to RUN.

   This causes the standby controller to be restarted and updated. It also initiates an automatic standby-master switchover and sets the "old" master to STOP. The "software modification" parameter is automatically reset.
6. Replace the "old" master's memory card.
7. Execute an overall reset and a cold restart with activation of the CPU 948R as standby.

The above sequence makes possible interruption-free software modification (changing of the user program) simply by replacing the memory card.

IMPORTANT

A bumpless switchover from standby to master can only be guaranteed if the use of the dynamic data is not changed.

9.7 Upgrading the CPU 948R's RAM or Version Number

Conditions and Upgrading Sequence

Upgrading of the CPU 948R's RAM capacity must be made known to the operating system via the "CPU 948R upgrade" parameter. The CPU's new user RAM capacity must be the same as, or higher than, the old one. The user program in the master CPU is transferred to the standby CPU. All data blocks in the master's RAM are transferred to the same locations in the standby's RAM.

The contents of the data blocks in the COM H list, timers, counters, flags and system data are transferred at one synchronization point. Upon completion of the activation phase, a standby-master switchover is initiated and the CPU which is now the standby stops. The "CPU 948R upgrade" parameter is reset.

Operator Input Sequence

Proceed as follows to upgrade the S5-155H without shutting down the system:

1. Using COM 155H, set the "PLC upgrade CPU 948R function" parameter in the master controller. This causes deactivation of Restart function "DB 0 comparison".
2. Set the standby controller to STOP.
3. Switch off the controller that is now at STOP.
4. Replace the CPU 948R in the standby controller with a CPU that has more RAM or a newer version number.
5. Switch the power back on.
6. Execute an overall reset.
7. Execute a cold restart by setting the standby's CPU to RUN.

   Step 7 restarts and updates the standby controller. The cold restart is followed by an automatic standby-master switchover, after which the "old" master stops. The "CPU 948R upgrade" parameter is automatically reset.
8. Replace the CPU 948R on the "old" master as described in steps 3, 4, 5 and 6.
9. Execute an Overall Reset and a cold restart with activation of the CPU 948R as standby.

Observing this sequence makes possible interruption-free upgrading of the RAM memory or of the CPU itself. Downgrading (installing lower-capacity RAM or a CPU with a lower version number) is not possible during operation.

Typical Applications

10

This chapter contains sample applications for various configurations of an S5-155H system. When you have worked through these examples, you will have an H system you can use for any purpose and expand to meet your specific requirements.

10.1 Task and Required Resources

Problem Definition

The steps making up the overall application should be carried out in the following order:

- First, you put the hardware into operation.
- Next, you add switched I/Os.
- Then you add one-sided I/Os to the switched I/Os.
- Finally, you add redundant I/Os with error locating facilities.

Hardware

The following hardware is required for our sample application:

- Two S5-135U/155U central controllers, each with a CPU 948R
- One IM 324R interface module
- Three IM 304 interface modules
- Three 721 interface cables
- One EU 185U expansion unit
- Two IM 314R interface modules
- Three 430 digital input modules
- Two 451 digital output modules
- Two 760-0HA11 terminators

Software

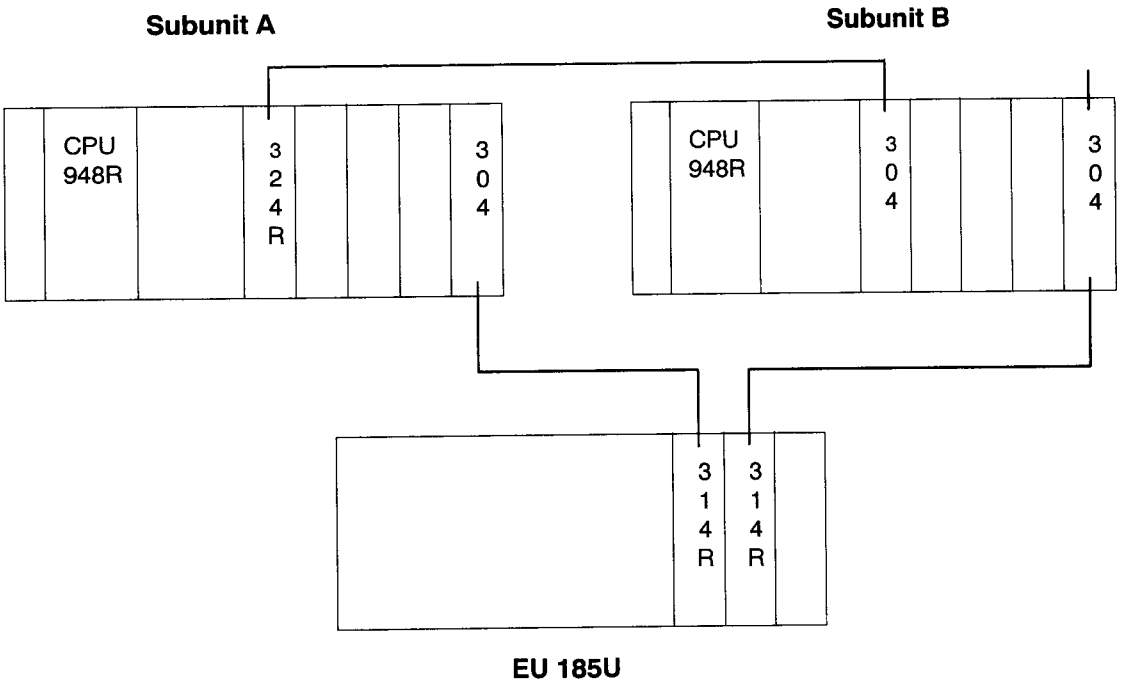
You require the following software:

- COM 155H (version 3.0 or newer)
- STEP 5 basic package, level 5

10.2 Installing the Hardware

System Configuration

You are going to install and configure an S5-155H system as shown in the diagram below.



Establishing a CC-to-CC Parallel Link

The purpose of this step is to establish the parallel link between the two CCs via the IM 324R and IM 304 (6ES5 304-3UB11) interface modules.

No changes to the jumper settings on the IM 324R are necessary (see Section 4.2). You must change the jumper settings on the 6ES5 304-3UB.. as shown in the diagram on the next page.

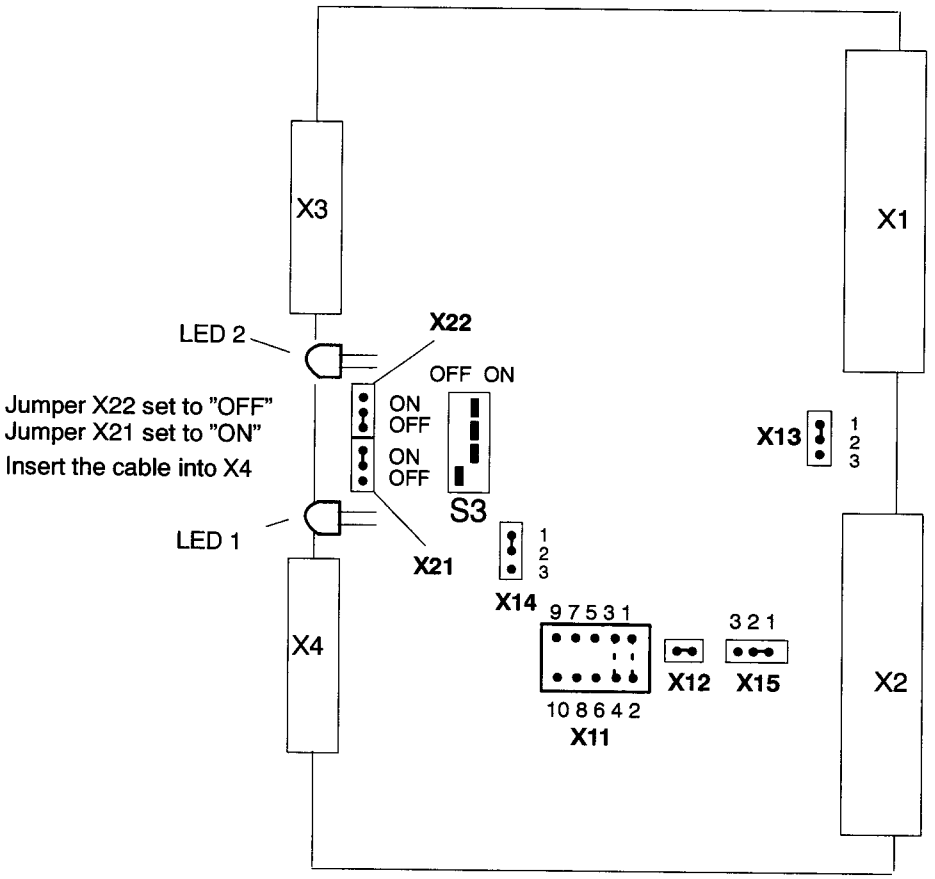
Plug the IM 324R into slot 131 in one of the subunits. This subunit will be referred to as subunit **A**.

10

Jumper Settings on the IM 304

Configure the jumpers on the IM 304 (module 6ES5 304-3UB11) as shown in the diagram below.

The setting for X11 may not exceed 100 m.



X11: Adaption to different cable lengths

	Jumper plug X11				
Jumper position	9 7 5 3 1 *) 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2
Cable length	Max. 10 m	10 to 100 m			

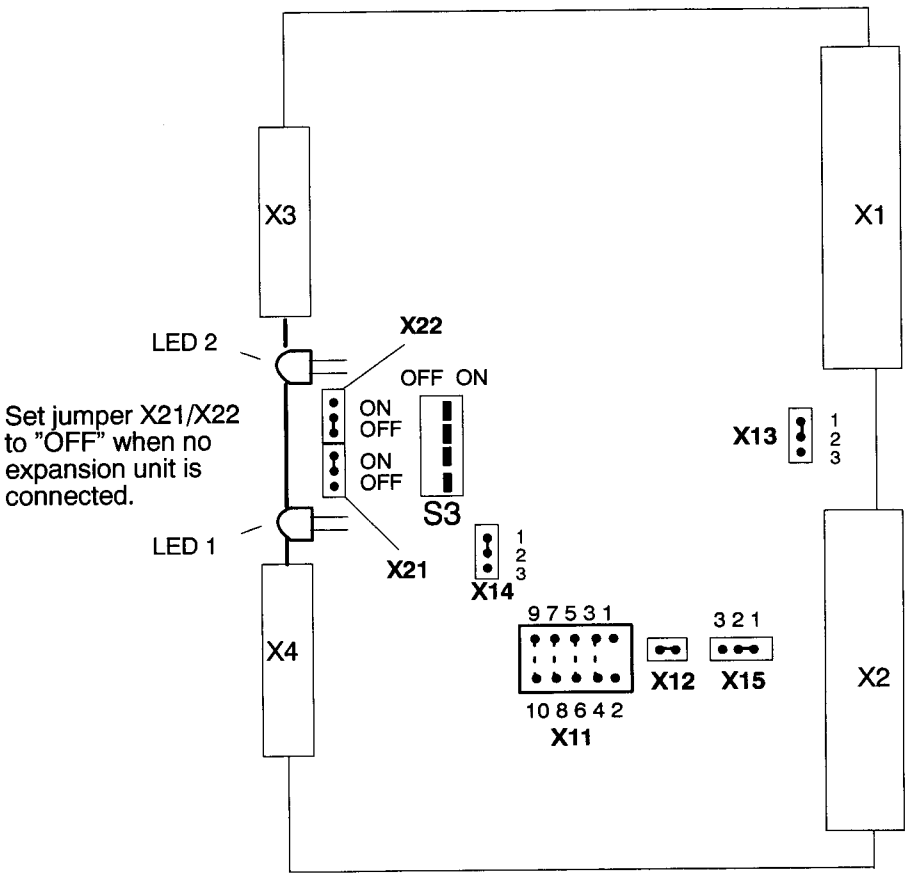
*) These settings are only allowed for interfacing the IM 304 and IM 324R in an S5-155H system. The length of the link to interface X4 determines the position of jumper X11.

Installing the I/O Bus

The purpose of this step is to establish the symmetrical link between the CC and the EU via the IM 304 (6ES5 304-3UB11) in subunits A and B and the two IM 314Rs in the expansion unit.

Jumper Settings on the IM 304

Set the jumpers on the IM 304 (module 6ES5 304-3UB11) as shown in the diagram below.



X11: Adaption to different cable lengths

	Jumper plug X11				
Jumper position	9 7 5 3 1 *) 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2	9 7 5 3 1 10 8 6 4 2
Cable length		1 to 100 m	100 to 250 m	250 to 450 m	450 to 600 m

*) These settings are permitted only for interfacing the IM 304 and the IM 324R in an S5-155H system.

The length of the link to interface X3 or X4 determines the position of jumper X11.

10.3 Configuring Switched I/Os

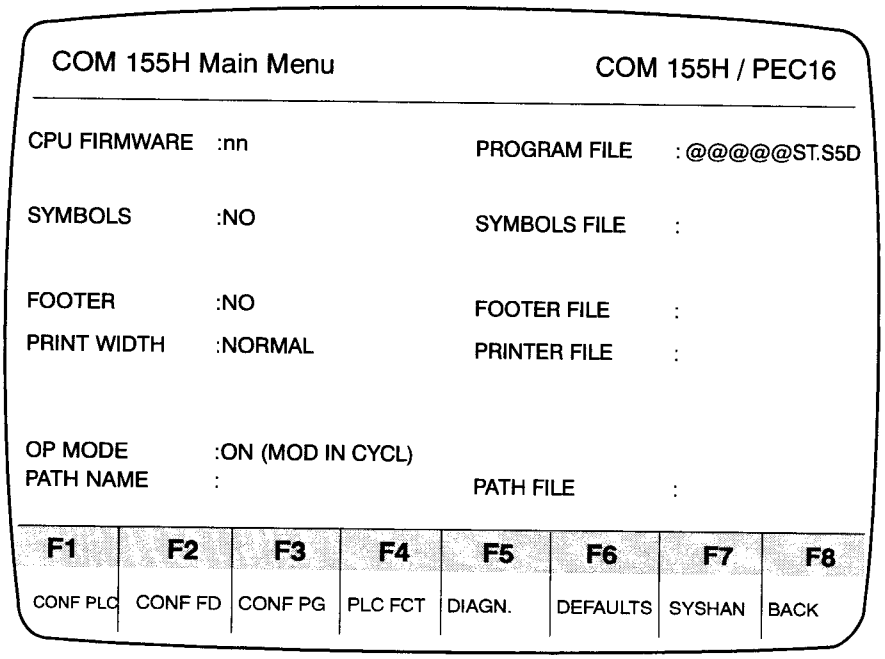
This section shows you how to configure four output bytes (bytes 8 to 11) and three input bytes (bytes 8 to 10) in switched I/Os.

1. Insert the input and output modules with the relevant settings (DI = address 8, DQ = address 8) and readback module into the EU 185U. Connect your programmer to the CPU in subunit A.

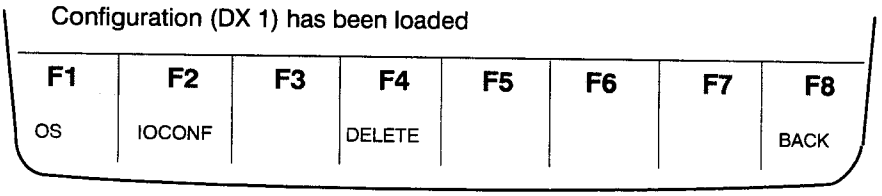
Calling COM 155H

2. Call up the Package Selection form on the programmer monitor with "S5". Position the cursor to the "COM 155H" line and press function key F1 to call the COM 155H programming software.
3. After entering the name of the program file and choosing the "ON" mode, press F6 <EXEC>.

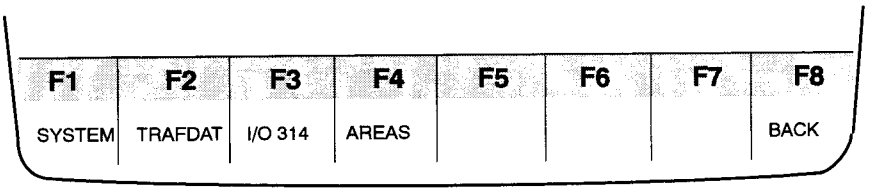
The COM 155H "Main Menu" form for STEP 5, level 5, appears on the screen.



4. Press function key F2 <CONF FD> to select the S5-155H configuring form.

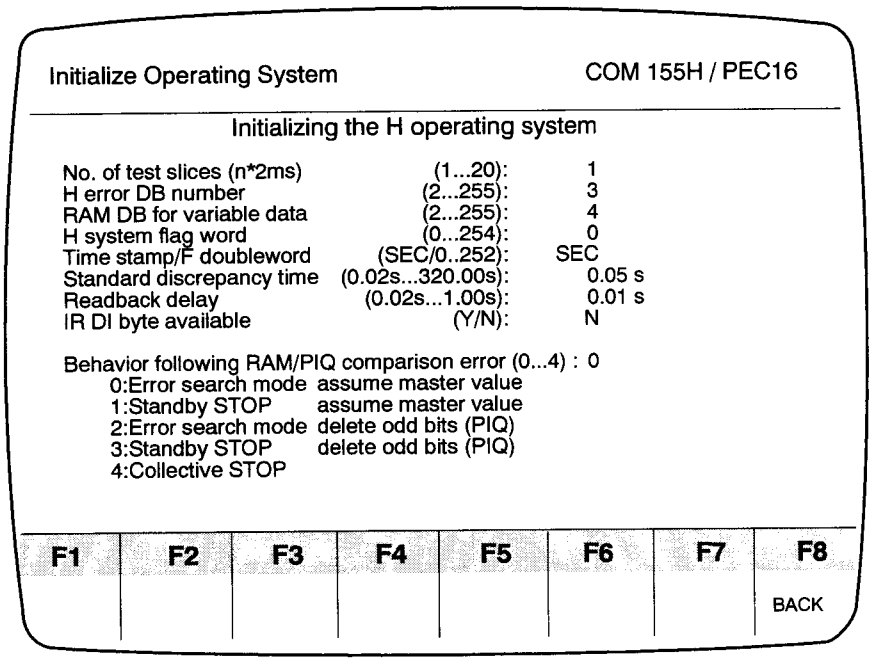


5. Press function key F1 <OS> to screen the softkey menu for "Initialize Operating System".



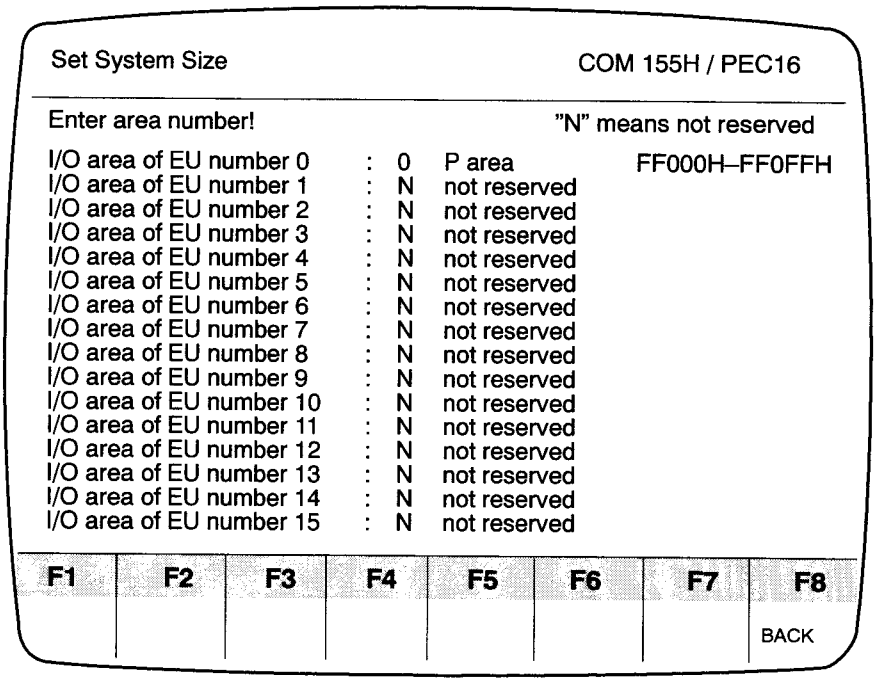
6. Press function key F1 <SYSTEM>.

Initializing the Operating System (COM 155H)

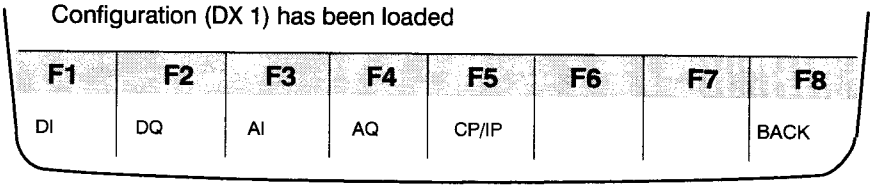


7. Enter "198" as H flag word. Confirm the default values for all other parameters. Press F8 <BACK> to save the parameters and return to the "Initialize operating system" softkey menu.
8. Press function key F3 <I/O 314>.

Setting the System Size

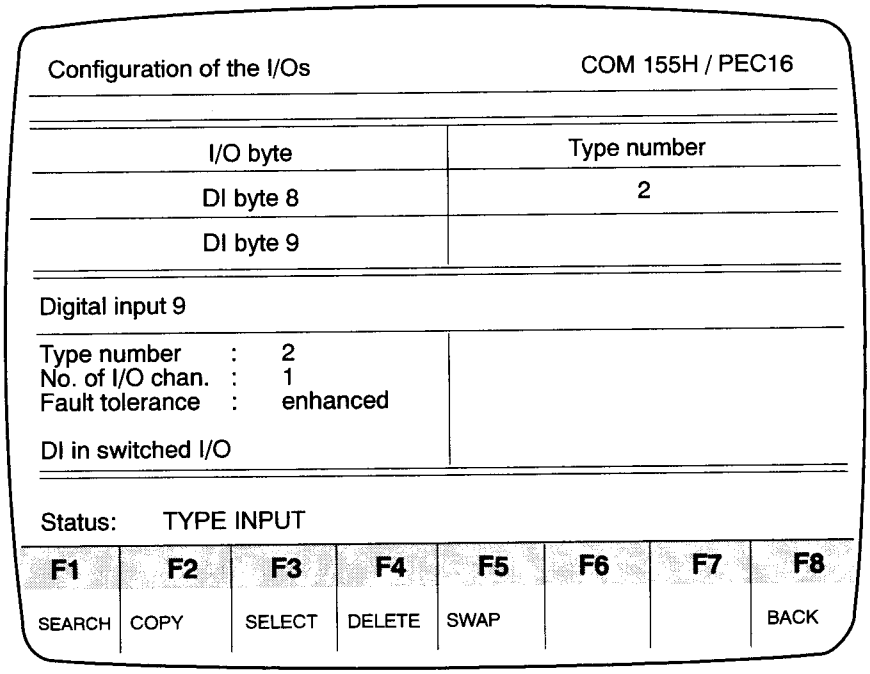


9. Since your expansion unit (EU 0) is to be operated in the P area, you must enter area number "0" (for P area).
10. Press function key F8 <BACK> to return to the COM 155H main menu. Press function key F2 <IOCONF> to select the softkey menu for "Configuration of the I/Os".



11. Press function key F1 <DI>.

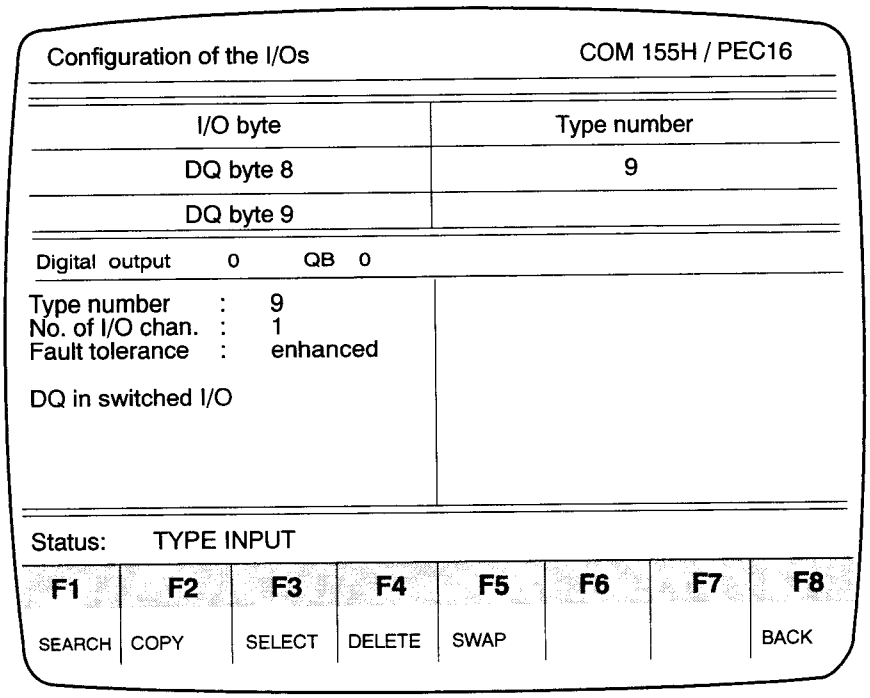
Configuring Digital Inputs



12. Enter the type number "1" (DI in switched I/O) for bytes 8 to 10, then press function key F8 <BACK>. Finally, press function key F2 <DQ> to enter the "Configuration of the I/Os" screen.

10

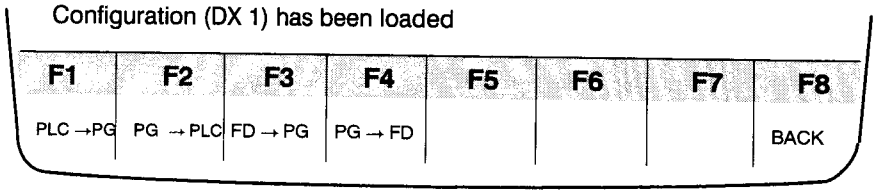
Configuring Digital Outputs



13. Enter type number "9" (DQ in switched I/O) for bytes 8 to 11.
14. Press function key F8 <BACK> three times in succession to select the "OVERWRITE CONFIGURATION (DX 1) ON FLOPPY?" prompt. Press <INSERT> to return to the COM 155H main menu.

Transferring Configuring Data (DX 1)

15. Press function key F7 <SYSHAN>, then function key F3 <TRAN/LOAD> to select the softkey menu for "I/O transfer/load".



16. Transfer the new DX 1 with F2 <PG → PLC> to your subunit A.
17. Press function key F8 <BACK> three times in succession to exit COM 155H.

Starting Up the PLC

Your configuring data (DX 1) is now in subunit A.

18. Execute a cold restart on subunit A. After completion of the self-test (red and green LEDs are on), the CPU enters the RUN mode (green LED is on).

    The operating system generates the configured error data block and RAM-DB automatically.

19. Execute a cold restart on subunit B (after an overall reset has been performed).

    The program in subunit A (master) is transferred to subunit B, and subunit B is "activated". The red and green LEDs on subunit B flash alternately. Following termination of the self-test (red and green LEDs show a steady light), the standby's CPU also enters the RUN mode. A flashing green LED shows which subunit is the standby.

You can now write programs just as you did for the S5-155U.

Note:
In the standby controller, direct I/O access operations must be suppressed during startup with the aid of the H flag word (status byte).

On-Line Functions

In Redundant mode, all write operations are executed on both PLCs simultaneously. Read operations in Redundant mode are the same as in the U system.

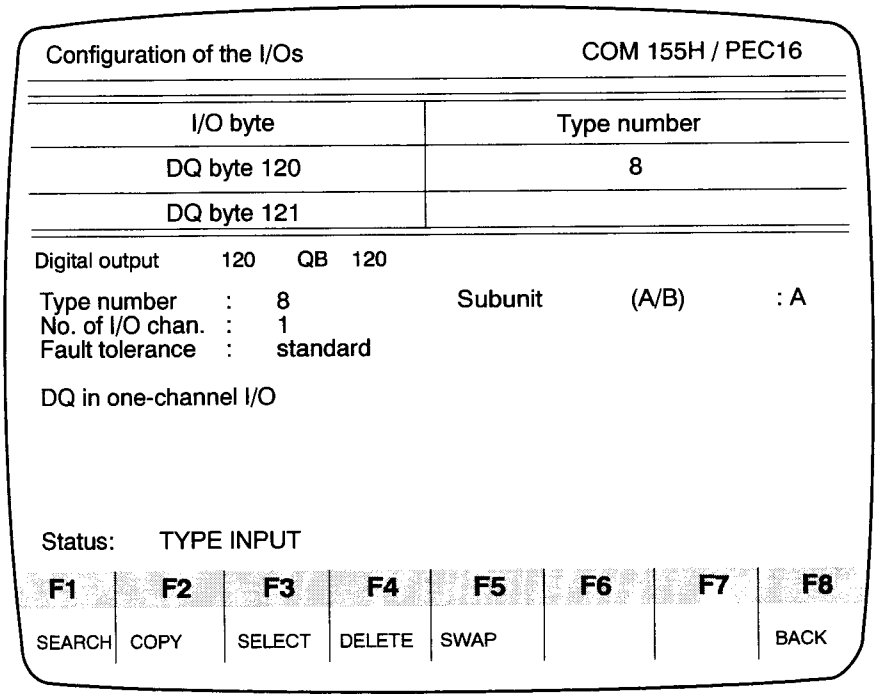
10.4 Configuring One-Sided I/Os

In this step, two output bytes (bytes 120 and 121) will be configured as one-sided I/Os in subunit A.

1. Insert the output module with the relevant setting (DQ = address 120) and readback module into subunit A.
2. Press function key F2 <CONF FD> to load DX 1 into the programmer.

Configuring Digital Outputs

3. Press F2 <IOCONF>, then F2 <DQ> to select the "Configuration of the I/Os" screen so that you can configure your one-sided digital outputs.



4. Enter type number "8" (DQ in one-channel I/O) and subunit "A" for bytes 120 and 121.
5. Press function key F8 <BACK> three times in succession to return to the main menu. Transfer the new DX 1 to the PLC.

Starting Up the PLC

Write your programs as you would for an S5-155U. Make sure, however, that the output bytes are assigned to only one subunit. When that subunit fails, the I/O bytes assigned to it are no longer available.

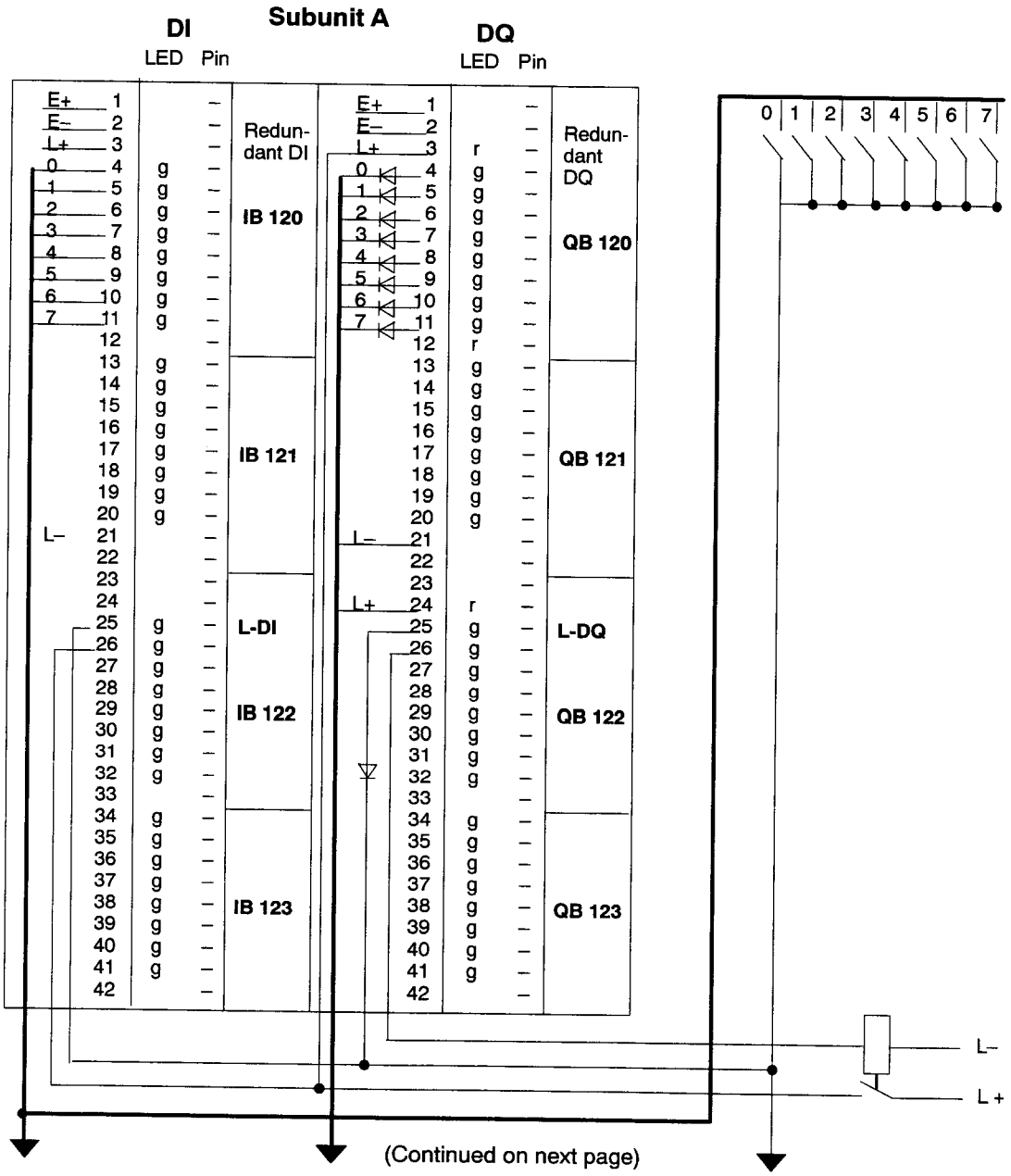
10

10.5 Configuring Redundant I/Os

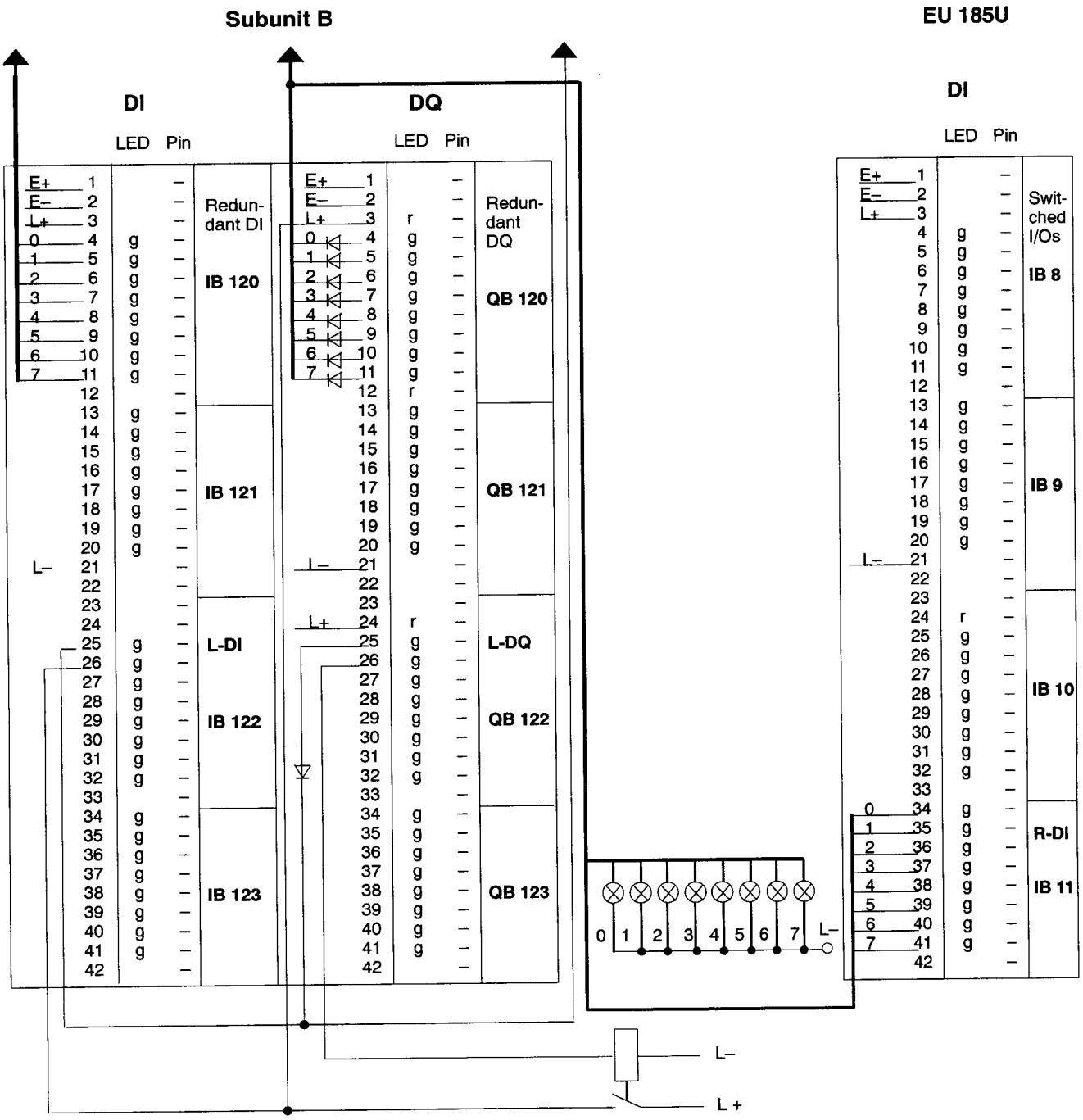
In this step, or section, which is regarded as a "separate project", as it were, you are going to configure one redundant input byte and one redundant output byte, each with error locating facility.

Interconnecting I/Os

1. Insert one 430 input module and one 451 output module with the module address 120 into each of the subunits. Then plug a 430 input module with the module address 8 into the switched I/O (expansion unit EU 185U).
2. Interconnect the modules as shown in the diagram below.



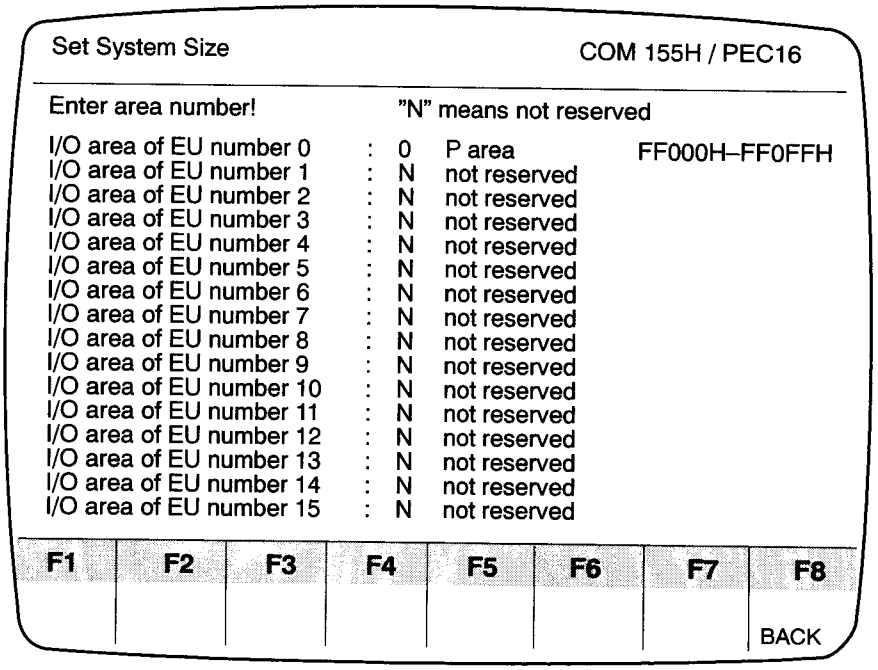
(Continued on next page)


Subunit B
EU 185U
DI
LED Pin
DQ
LED Pin
DI
LED Pin
E+ 1
E– 2
L+ 3
0 4
1 5
2 6
3 7
4 8
5 9
6 10
7 11
12
13
14
15
16
17
18
19
20
L– 21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
Redun-dant DI
IB 120
IB 121
L-DI
IB 122
IB 123
Redun-dant DQ
QB 120
QB 121
L-DQ
QB 122
QB 123
Swit-ched I/Os
IB 8
IB 9
IB 10
R-DI
IB 11
0 1 2 3 4 5 6 7 L–
L–
L+

10

Setting the System Size

3. If you have not yet entered expansion unit EU 0 with area number "0" (P area), do so now.

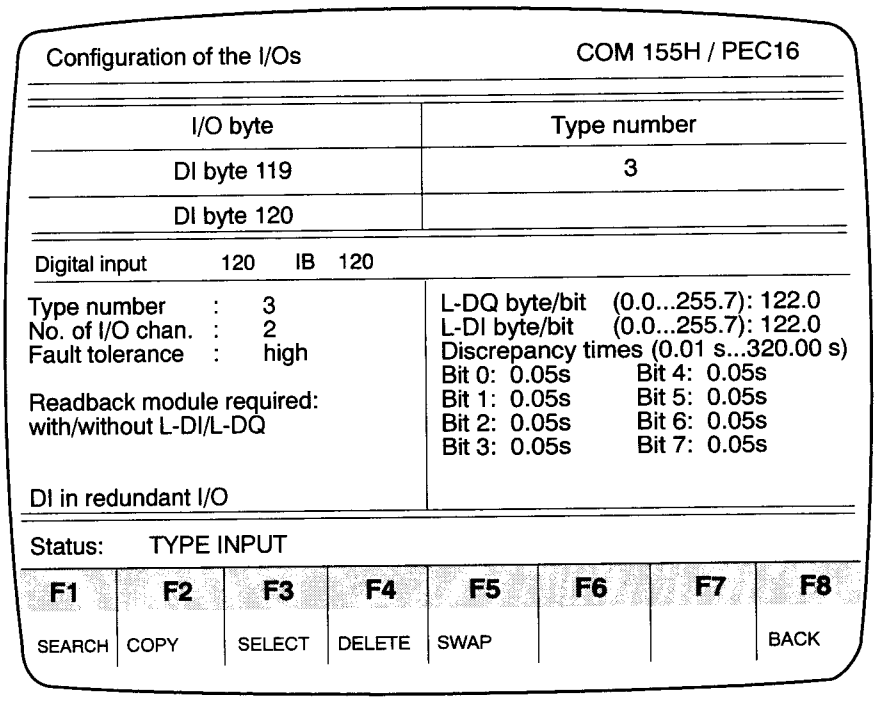


4. Press function key F8 <BACK> twice in succession to return to the COM 155H configuring menu.

5. Press function key F2 <IOCONF>, then function key F1 <DI> to select the "Configuration of the I/Os" screen.

Configuring Redundant Inputs

6. Enter type number "3" (DI in redundant I/O) for byte 120. Enter "122.0" as L-DQ and as L-DI bit. Confirm the default values for the discrepancy times.

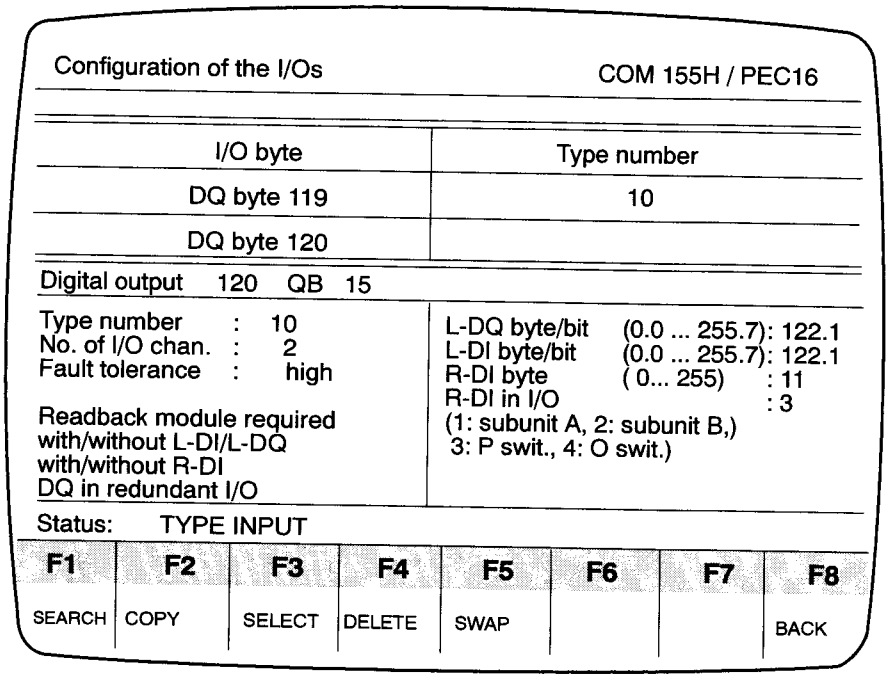


7. Press function key F8 <BACK>, then function key F2 <DQ> to select the "Configuration of the I/Os" screen.

Configuring Redundant Outputs

8. Enter type "10" (DQ in redundant I/O) for byte 120. Enter "122.1" as L-DQ bit and L-DI bit. Enter byte "11" in switched I/O ("3") as R-DI (readback digital input).

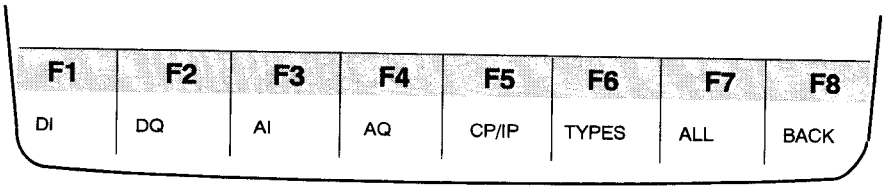
10



9. Press function key F8 <BACK> twice in succession to return to the COM 155H main menu.
10. Transfer the new DX 1 to the PLC.

Printing out the Configuring Data

11. In the main menu, press function key F7 <SYSHAN>, then F4 <PRINT> to select the "Print menu" screen.



12. Press F1 <DI> and F2 <DQ> for the following printouts of your I/O modules.

Digital Inputs/Outputs

I/O Byte	Symbol	Type	Subunit		I/O Byte	Symbol	Type	Subunit
IB 8		2			IB 9		2	
IB 10		2			IB 11		R-DI	
IB 120		3			IB 122		L-DI	
QB 120		10			QB 122		L-DQ	

10.6 Redundant Point-To-Point Link

To establish a redundant link, you can plug the relevant CP either

- into both central controllers (for two-channel redundancy) or
- into switched expansion unis (for switched redundancy).

The H-specific special features of a two-channel redundant and of a switched redundant point-to-point connection are discussed using the CP 525 as an example.

Establishing links with other CPs, such as CP 523, CP 524, CP 530, CP 544 and CP 143, is done in the same way.

Two-Channel Redundant Point-to-Point Link

The following hardware is required:

- One S5-155H in its basic configuration
- One S5-115U with power supply unit, CPU and IM 306 as link partner
- Four CP 252s

The CP 525 in the S5-155H's subunit A is referred to as CP A_H, the CP 525 in subunit B as CP B_H. The two corresponding CPs in the S5-115U are referred to as CP A_U and CP B_U.

The system is illustrated in Figure 10-1.

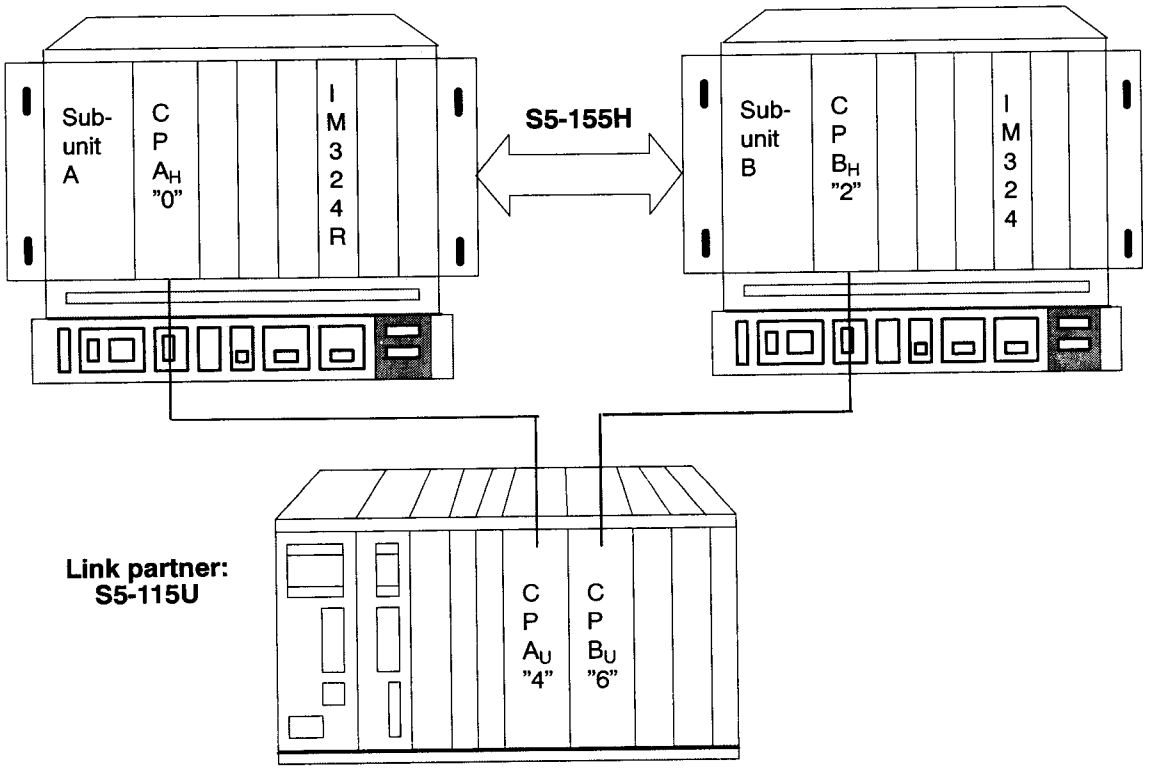


Figure 10-1 Schematic of a Two-Channel Redundant Point-to-Point Link

You must set the addresses over which the CPUs can communicate with their CPs on DIP switches located on the rear of the CPs.

- Set interface number (SSNR) 0 on CP A_H; CP A_H thus reserves pages 0 and 1.
- CP B_H is assigned page numbers 2 and 3, that is, SSNR = 2.
- CP A_U is assigned SSNR 4, CP B_U SSNR 6.

Initializing the CPs with COM 155H and COM 525

Using COM 155H, you must initialize CP A_H in subunit A and CP B_H in subunit B as one-sided CPs, and not as redundant CPs.

The COM 525 parameter initialization software is used to define how the CPs are to function. The two CPs in the U controller are configured only with the COM 525 parameter initialization software in the usual manner.

Programming

H-specific parts of the user program are:

- Restart routine
- Data interchange and
- OB 1.

These are discussed in the following sections.

Restart PLC

Because both subunits have the same user program, but the two CPs for the redundant link (CP A_H and CP B_H) have different page numbers, the SYNCHRON data handling block (DHB) must be invoked conditionally for CP A_H (:JC FB 125) in subunit A.

The same applies to CP B_H in subunit B. Because of SYNCHRON, the CPU knows that the configuration includes a CP.

If the SYNCHRON DHB is not invoked conditionally in both subunits, an error is entered in the PAFE byte (in this case in flag byte 198 or 199).

This "feature" must be programmed in a restart FB (see Figure 10-2). You must invoke this restart FB in all restart OBs (OB 20, OB 21 and OB 22) so that it will be processed no matter what kind of restart is involved, thus avoiding errors.

STL	Explanation
FBxx	
NAME :ANLAUF	
:A F 0.4	CC is subunit A
:JC FB 125	
NAME :SYNCHRON	
SSNR :KY 0.0	Page no. for CP A_H
BLGR :KY 0.6	
PAFE :FY 198	
:AN F 0.4	CC is subunit B
:JC FB 125	
NAME :SYNCHRON	
SSNR :KY 0.2	Page no. for CP B_H
BLGR :KY 0.6	
PAFE :FY 199	
:BE	

Figure 10-2 Restart FB for S5-155H

Data Interchange

Data can be interchanged in two directions:

- S5-155H → S5-115U and
- S5-155H ← S5-115U

10

Data Interchange S5-155H →S5-115U

Depending on how reliable you want the interchange of data to be in a redundant link, you can check both CPs in the S5-115U controller to see if

- any data has arrived at all,
- the same amount of data has arrived in both, or
- the same data has arrived.

The more reliability you want, the more you have to program. In the example, we have kept to the simplest case, which is "whether any data has arrived at all" (see Figure 10-3). In contrast to a "simple" link, the user need only

- program Send requests for the CP A_H and CP B_H, and
- program the "Data test" FB.

FB 252 is the standard FB RECEIVE. For detailed information on FB 252, please refer to Catalog ST 57. On the S5-115U, FB 252 is integrated in the operating system.

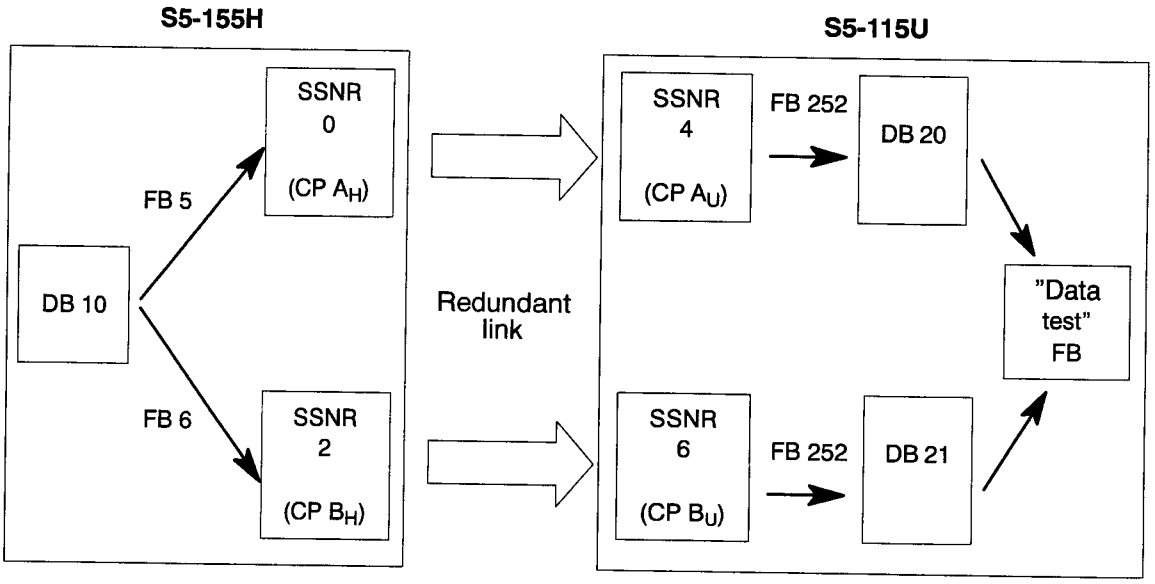


Figure 10-3 Schematic of Data Interchange S5-155H → S5-115U

Send Requests for CP A_H and CP B_H

The user must program SEND requests for CPs A_H and B_H which transfer specific data from DB 10 to the respective page. These requests must be programmed in a function block. In the example, FB 5 is used for CP A_H and FB 6 for CP B_H. In contrast to a standard computer link, the first word in DB 10 is used as frame counter. FB 5 and FB 6 must be called conditionally in the S5-155H's OB 1.

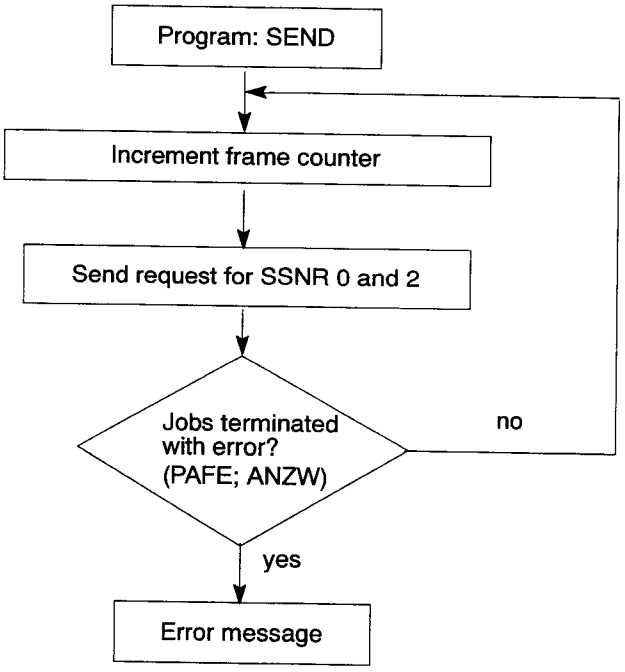


Figure 10-4 Structogram of FB 5 and FB 6

The "Data Test" FB

In the "Data test" FB, the frame counter in the first word of DBs 20 and 21 is used to ascertain whether the two U CPs have received any data at all. If a U CP is not receiving data, the "Data test" FB recognizes this fact and reports it. Figure 10-5 shows a structogram of the "Data test" FB.

You can program this FB both for S5-155H ← S5-115U and S5-155H → S5-115U.

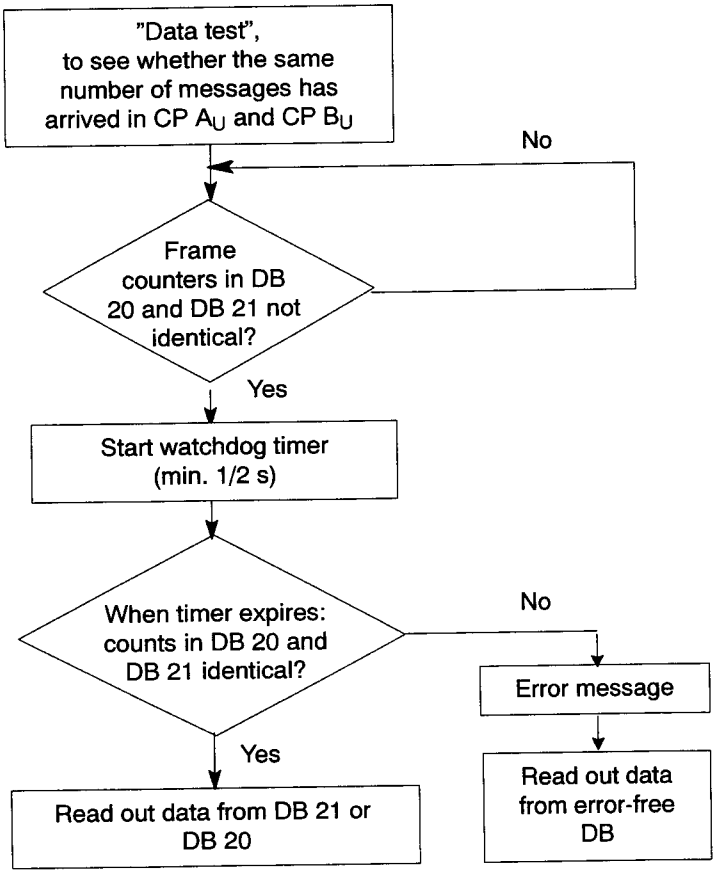


Figure 10-5 Structogram of the "Data Test" FB

Data Interchange S5-155H ← S5-115U

In this case, too, we have taken as an example the simplest case; that is, whether any data has arrived at all. As for S5-155H → S5-115U interchanges, the user must:

- program Send requests for CP A_U and CP B_U, and
- program the "Data test" FB

FB 127 is the standard FB RECEIVE. For detailed information on FB 127, see Catalog ST 57.

Send Requests for CP A_U and CP B_U

In the example (see Figure 10-6), the Send requests for CP A_U and CP B_U are in FB 15 and FB 16. The structograms for FBs 15 and 16 are the same as those for FBs 5 and 6 (see Figure 10-4). FB 15 and FB 16 must be called unconditionally in the S5-115U's OB 1.

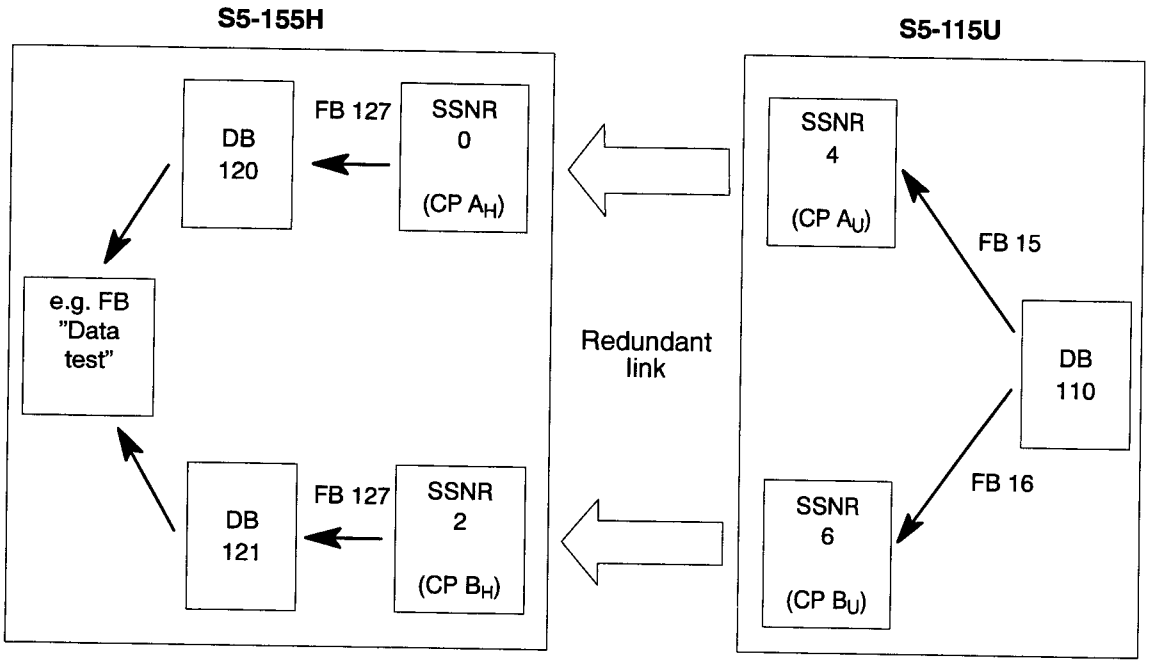


Figure 10-6 Schematic of Data Interchange S5-155H ← S5-115U

OB 1 in the S5-155H

FB 5 for CP A_H and FB 6 for CP B_H are called conditionally in the S5-155H's OB 1, depending on whether the S5-155H is in Redundant mode or subunit A or B is in Solo mode. Figure 10-7 shows you how you can program OB 1.

STL	Explanation
OB 1	
...	
:A F 0.1	PLC in Redundant mode?
:	PLC not in Redundant
:O F 0.4	mode and subunit A
:JC FB 5	in RUN mode?
NAME:SUBUNIT A	
:A F 0.1	PLC in Redundant mode?
:	PLC not in Redundant
:ON F 0.4	mode and subunit B
:JC FB 6	in RUN mode?
NAME:SUBUNIT B	
:BE	

Figure 10-7 Conditional Calling of the "Send Requests" in OB 1 of the S5-155H

Switched Redundant Point-to-Point Link

The following hardware is needed:

- One S5-155H in its basic configuration with two IM 304s, connecting cables and terminating resistor connectors (terminators)
- Two EU 185s with four IM 314Rs
- One S5-115U with power supply unit, CPU and IM 306 as link partner
- Four CP 525s.

  The CP 525 in EI 1 of the S5-155H (see Figure 10-8) is referred to as CP 1_H, the one in EU 2 as CP 2_H. The two corresponding CP 525s in the S5-115U are referred to as CP 1_U and CP 2_U.

  – You could also implement the link using one EU 185. Such a system would have a much lower degree of fault tolerance than one with two EUs, however, since the whole EU would have to be shut down to replace a CP.
- You must set the address (page numbers) over which the CPUs communicate with their CPs with DIP switches on the CP modules.
- Set interface number (SSNR) 0 on CP 1_H; CP 1_H thus reserves pages 0 and 1.
- CP 2_H is assigned page numbers 2 and 3; that is, SSNR = 2.
- CP 1_U is assigned SSNR 4, CP 2_U SSNR 6.

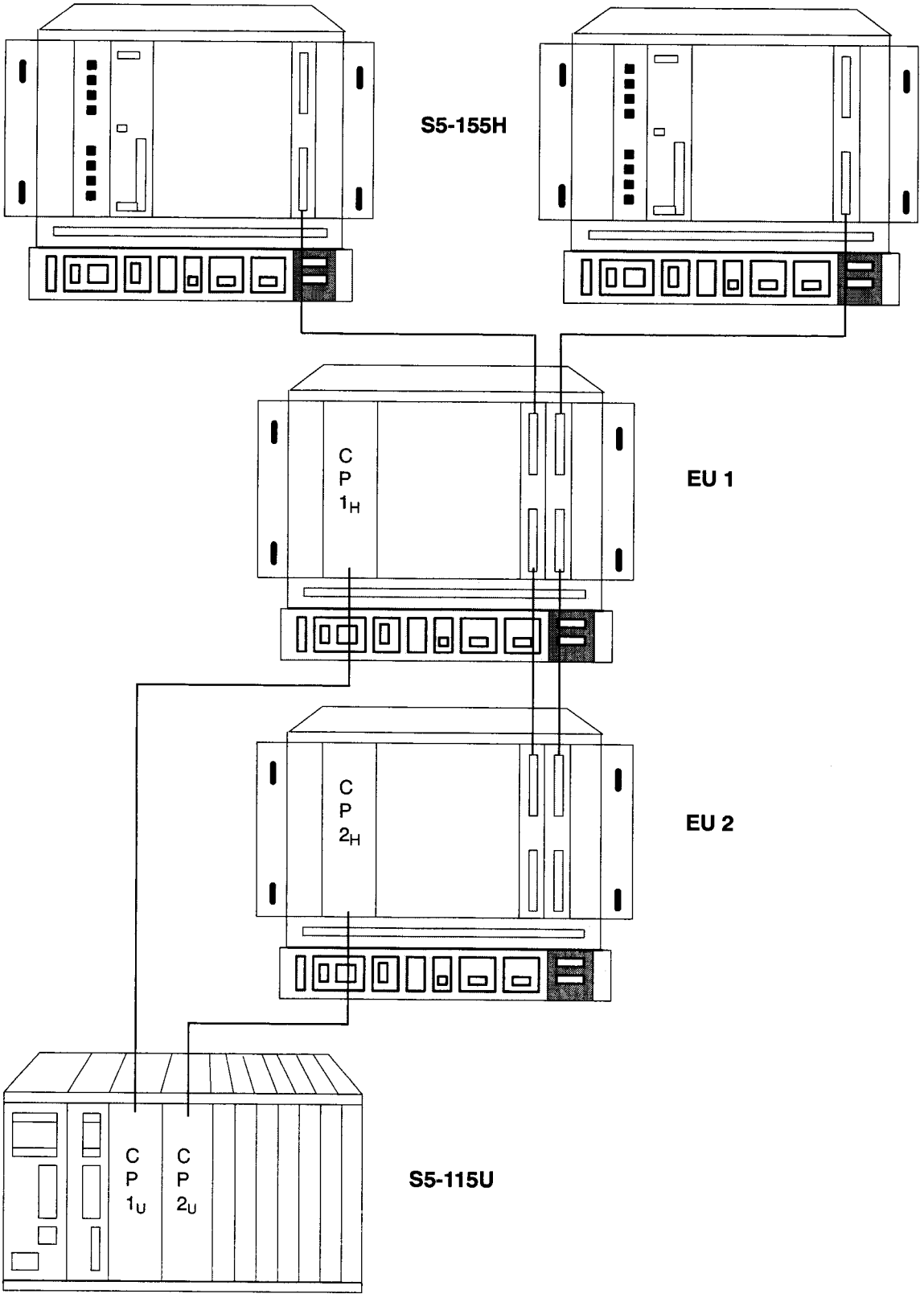


Figure 10-8 Schematic of a Switched Redundant Point-to-Point Link

Parameter Initialization with COM 155H and COM 525

Using COM 155H, CP 1_H and CP 2_H must be configured as switched CPs, not as redundant CPs. You must use the COM 525 parameter initialization software to define how the CPs are to function. The two CPs in the U-series controller are configured only with the COM 525 parameter initialization software.

Programming

H-specific parts of the user program are:

- Restart routine
- Data interchange
- OB 1 and
- Reactivation of a failed CP

PLC Restart

Both subunits have the same user program, but the two CPs for the redundant link (CP 1_H and CP 2_H) have different page numbers. Because CP 1_H and CP 2_H are plugged into switched EUs, the master subunit must be synchronized by invoking FB 125 (:JC FB 125).

If the standby controller also invokes the SYNCHRON data handling block, an error is entered in the PAFE byte (in this case flag byte 198 or 199). You must program this "feature" in a restart FB (see Figure 10-9). This FB must then be invoked in all restart OBs (OB 20, OB 21 and OB 22) so that it will always be processed, no matter what kind of restart is involved, thus avoiding errors.

STL	Explanation
FBx	
NAME :RESTART	
:A F 0.2	CC is master
:JC FB 125	
NAME :SYNCHRON	
SSNR :KY 0.0	Page no. for CP 1_H
BLGR :KY 0.6	
PAFE :FY 198	
:A F 0.2	CC is master
:JC FB 125	
NAME :SYNCHRON	
SSNR :KY 0.2	Page no. for CP 2_H
BLGR :KY 0.6	
PAFE :FY 199	
... :BE	

Figure 10-9 Restart FB for H Controller

Data Interchange

Data interchange is exactly the same as over the two-channel redundant point-to-point link (see above).

OB 1 in the S5-155H

FB 5 for CP 1_H and FB 6 for CP 2_H are invoked unconditionally in the S5-155H's OB 1 (see Figure 10-10). FBs 5 and 6 were discussed in detail in Section 10.6.

```
STL                          Explanation
OB 1
...
    :JU    FB 5              FB for CP 1H
...
...
    :JU    FB 6              FB for CP 2H
...
```

Figure 10-10 Absolute Call of the "Send Requests" in OB 1 of the S5-155H

Reactivating a Failed CP (In-Cycle Synchronization)

Should CP 1_H or CP 2_H fail, it must be reincluded in the process following its repair without a cold or warm CPU restart. This, of course, means that CP synchronization must take place in the current cycle, which in turn means that you must invoke FB (see Figure 10-11) unconditionally in OB 1.

```
STL                          Explanation
    :                        Program for CP 1
    :O F 10.0                F 10.0 must be set when
    :LKT 150.2               CP 1 has been repaired
    :SE T 1                  Depassivation takes max. 15 sec.
    :O(
    :L    FY 8               PAFE byte for CP 1
    :L KH 0000
    :><F

    :)                       F 10.0 must be set when
    :O F  10.0               CP 1 has been repaired
    :R F  10.0
    :=F   11.0

    :JC   FB 125
NAME:SYNCHRON
SSNR:     KY0.0              Page no. for CP 1
BLGR:     KY0.6
PAFE:     FY 8
    :A F  11.0               Depassivation requested?
    :A T  1                  15 sec. elapsed?
    :BEC                     If yes, -> BE, as only one CP can be
    :                        activated at a time
    :A F  11.0               Have 15 sec. elapsed?
    :R F  11.0
    :JC   FBxx               Error CP 1 cannot be
    :L KB 0                  depassivated
    :T FY 8
```

Figure 10-11 FB for Reactivating a Failed CP (continued on next page)

STL		Explanation
:		
:		**Program for CP 2**
:A F	10.1	F 10.1 must be set when
:L KT	150.2	CP 2 has been repaired
:SE T	1	Depassivation takes max. 15 sec.
:O(		
:L	FY 9	PAFE byte for CP 2
:LKH	0000	
:><F		
:)		
:A F	10.1	F 10.1 must be set when
:R F	10.1	CP 2 has been repaired
:=F	11.1	
:JC	FB 125	
NAME:SYNCHRON		
SSNR:	KY0.2	Page no. for CP 2
BLGR:	KY0.6	
PAFE:	FY 9	
:A T	1	Only one CP may be activated at a time
:BEC		
:		
:A F	11.1	
:R F	11.1	Error CP 2 cannot be depassivated
:JC	FBxx	
:L KB	0	
:T FY	9	**Program for CP 3**
:		
:		

Figure 10-11 continued

Technical Specifications: IM 314R / IM 324R

11

This chapter contains the technical specifications of the IM 314R and IM 324R interface modules.

11.1 Technical Specifications of the IM 314R Interface Module

The electronics are accommodated on a printed-circuit board in double-height Eurocard format. There are two 48-pin connectors of range 2 to connect the module to the S5 bus of the expansion unit. The frontplate of the module is 1 1/3 standard slots wide and has two 50-pin D-type connectors for the symmetrical cable (6ES5 721-0xxx0) and four LEDs.

Supply voltage:	+ 5 V ± 5%
Current consumption:	approx. 900 mA
Timeout for internal registers:	approx. 10 µs
Max. cable length from IM 304 to the last IM 314R	600 m
Max. potential difference between H components (equipotential bonding conductor to design regulations):	5 V
Max. number of IM 314Rs on the bus:	4
Weight:	approx. 350 g
Dimensions:	160 mm x 233.4 mm
Width of frontplate:	20.32 mm
Front connector: connectors	Two 50-pin male
Backplane connector:	Two 48-pin ES 902, range 2

Environmental Conditions

Operating temperature range: (Operation without fan permissible, but natural convection must be guaranteed)	0 °C to 55 °C (32 °F to 131 °F)
Non-operating temperature:	– 40 °C to + 70 °C (– 40 °F to + 158 °F)
Max. relative humidity:	95 % at 25 °C (77 °F), no condensation
Max. operating altitude:	3500 m above sea-level

Connector Pin Assignment

Pin		Pin		Pin	
1	Shield	18	+AD 8	34	+MEMR/
2	+AD 12	19	- AD 8	35	- MEMR/
3	- AD12	20	+AD 9	36	+MEMW/
4	+AD 13	21	- AD 9	37	- MEMW/
5	- AD13	22	+AD 10	38	+ALE
6	+AD 14	23	- AD10	39	- ALE
7	- AD14	24	+AD 11	40	+BASP
8	+AD 15	25	- AD11	41	- BASP
9	- AD15	26	+AD 3	42	+AD 0
10	+AD 6	27	- AD 3	43	- AD 0
11	- AD 6	28	+AD 4	44	+AD 1
12	+AD 7	29	- AD 4	45	- AD 1
13	- AD 7	30	+AD 5	46	+AD 2
14	+PEU	31	- AD 5	47	- AD 2
15	- PEU	32	+ZGU	48	+RDY/
16	P'	33	- ZGU	49	- RDY/
17	Shield			50	Ground

Figure 11-1 IM 314R: Pin Assignment of the X3 and X4 Front Connectors

Pin	d	b	z
2	Shield	M	+5 V
4		PESP	
6	A12	A 0	CPKL/*RESET
8	A13	A 1	MEMR/
10	A14	A 2	MEMW/
12	A15	A 3	RDY/
14		A 4	D 0
16		A 5	D 1
18		A 6	D 2
20		A 7	D 3
22		A 8	D 4
24		A 9	D 5
26		A10	D 6
28		A11	D 7
30	M	BASP	
32	BASPA/	M	M

Figure 11-2 IM 314R: Pin Assignment of the X1 Backplane Connector

Pin	d	b	z
2		M	+5 V
4		SA 0	NA 0
6	(CPKL/)	SA 1	NA 1
8		SA 2	NA 2
10		SA 3	NA 3
12	+5 V		
14	+5 V		
16			
18		NAU/	
20			
22		M	M
24			
26			
28			
30			
32		M	M

Figure 11-3 IM 314R: Pin Assignment of the X2 Backplane Connector

11.2 Technical Specifications of the IM 324R Interface Module

The electronics are accommodated on a printed-circuit board of double-height Eurocard format. There are two 48-pin connectors of range 2 to connect the module to the S5 bus of the expansion unit. The frontplate of the module is 1 1/3 standard slots wide and has two 50-pin D-type connectors for the symmetrical cable (6ES5 721-....). The frontplate also has a green LED and a connector for module replacement during operation (NON-STOP mode). The module also contains MOS components that are sensitive to electrostatic charge:

Supply voltage: + 5 V ± 5%	
Current consumption:	approx. 1000 mA
Memory capacity in the S5-155H:	4 x Kwords
Memory capacity in the S5-115H:	4 x Kwords
Max. access time (with one-sided RAM access:	100 ns
Response time (S5 bus): *)	approx. 300 ns
Response time (PK bus): *)	approx. 500 ns
Max. cable length from IM 304 to IM 324R:	100 m
Weight:	approx. 350 g
Dimensions:	160 mm x 233.4 mm
Front plate width:	20.32 mm
Front connector (X4, bottom):	50-pin, male connector
Backplane connector:	Two 48-pin ES 902, range 2
Frontplate connector, top (for power supply when replacing modules)	24 VDC ± 7 V 14 mA ± 6 m

Environmental Conditions

Operating temperature range: (Operation without fan permissible, but natural convection must be guaranteed)	0 °C to 55 °C (32 °F to 131 °F)
Non-operating temperature:	– 40 °C to + 70 °C (– 40 °F to + 158 °F)
Max. relative humidity:	95% at 25 °C (77 °F), no condensation
Max. operating altitude:	3500 m above sea-level

*) The response time is the time between the falling edge of the memory read (MEMR) or memory write (MEMW) signal at the IM 324R input and the falling edge of the ready (RDY) signal generated by the IM 324R, provided the RAM is not accessed by the other subunit.

Connector Pin Assignment

Note: The signals in parentheses are not transmitted via the interface.

Pin		Pin		Pin	
1	Shield	18	+AD 8	34	+MEMR/
2	+AD 12	19	- AD 8	35	- MEMR/
3	- AD 12	20	+AD 9	36	+MEMW/
4	+AD 13	21	- AD 9	37	- MEMW/
5	- AD 13	22	+AD 10	38	+ALE
6	+AD 14	23	- AD 10	39	- ALE
7	- AD 14	24	+AD 11	40	+BASP
8	+AD 15	25	- AD 11	41	- BASP
9	- AD 15	26	+AD 3	42	+AD 0
10	+AD 6	27	- AD 3	43	- AD 0
11	- AD 6	28	+AD 4	44	+AD 1
12	+AD 7	29	- AD 4	45	- AD 1
13	- AD 7	30	+AD 5	46	+AD 2
14	+(PEU)	31	- AD 5	47	- AD 2
15	- (PEU)	32	+ZGU	48	+RDY/
16	p'	33	- ZGU	49	- RDY/
17	Shield			50	Ground

Figure 11-4 IM 324R: Pin Assignment of the X4 Front Connector

Pin	d	b	z
2	Shield	M	+5 V
4			
6	A12	A 0	CPKL/*RESET
8	A13	A 1	MEMR/
10	A14	A 2	MEMW/
12	A15	A 3	RDY/
14		A 4	D 0
16		A 5	D 1
18		A 6	D 2
20		A 7	D 3
22		A 8	D 4
24		A 9	D 5
26		A10	D 6
28	DSI/	A11	D 7
30		BASP	
32	BASPA/	M	

Figure 11-5 IM 324R: Pin Assignment of the X1 Backplane Connector

Pin	d	b	z
2		M	+5 V
4		D 8	D 12
6	M	D 9	D 13
8		D 10	D14
10		D 11	D 15
12			
14			NAU/
16			
18			
20			
22			
24			
26			
28			
30			
32		M	

Spare Parts Coding plug C79334-A3011-B11

11.3 Readback Delays

Digital Input Modules

6ES5 420	:	30 ms each
6ES5 430	:	
6ES5 431	:	
6ES5 434	:	
6ES5 435	:	
6ES5 436	:	

Digital Output Modules

6ES5 441	:	30 ms each
6ES5 451	:	
6ES5 453	:	
6ES5 454	:	
6ES5 455	:	
6ES5 456	:	
6ES5 457	:	
6ES5 458	:	
Total readback delay		60 ms

For redundant 220 V digital modules, the readback delay must be at least 100 ms.

When using the ET 100U/ET 200 U, the readback delay must be at least $2t_{h\ max.}$. Please refer to the manual "ET 100U/ET 200U Distributed I/O System" for methods of calculating the value $t_{h\ max.}$

Glossary

12

The Glossary, whose contents are listed in alphabetical order, defines the most important 155H-specific terms and functions. Use the keyword index to find additional information on all terms and topics in the main body of the manual.

Configuring Block DX 1

The system program stores all configuring data entered with COM 155H during a configuring session at the programmer in data block DX 1. For this reason, DX 1 may be used for no other purpose.

An S5-155H can operate only when DX 1 has been loaded.

Depassivation

'Depassivation' is the opposite of 'passivation', and therefore revokes the former. The tests are also reactivated. Following repairs (module replacement), depassivation must be initiated.

Discrepancy Value

For a redundant analog input, you may configure a value to serve as the maximum amount by which analog value A (subunit A) may differ from analog value B (subunit B). The 'discrepancy value' is composed of an absolute portion and a relative portion. Only when this value is exceeded (and the configured discrepancy time has elapsed) does the 155H system program recognize an error.

Discrepancy Time

Redundant digital or analog inputs can show different signal states or input values within a relatively short space of time. The 'discrepancy time' is the time period during which the 155H system program will tolerate different signal states or input values. The default discrepancy time is 0.05 s, but the user may set it to a value between 0.01 and 320 s. The system program does not recognize an error until this time has elapsed.

Error Locating

Because of a series of comprehensive self-tests, the 155H system program cannot only detect errors and faults quickly, but can also locate them. With the aid of this facility, it is possible to find out which modules are defective and must be replaced.

A special error locating facility (or LF) must be configured for digital I/Os which are to be operated on a NON-STOP basis. As soon as a defective module has been located, the 155H system program automatically shuts that module down and operation from that point on is one-sided.

Error Search Mode

'Error Search' mode is invoked when the 155H system program finds discrepancies between the RAM contents or process images in master and standby. In this mode, the master scans the cyclic user program while the standby executes the self-test to locate the problem.

Group Power Supply

A 'group' constitutes all sensors for redundant digital inputs (DIs) or digital outputs (DQs) that are served by the same locating digital output (L-DQ). The smallest possible group consists of **one** redundant DI or DQ byte, the largest possible group of **all** redundant DIs or DQs in an S5-155H controller.

H Flag Word

The user specifies the number of the 'H flag word' during his configuring session with COM 155H. This flag word comprise a status byte and a control byte. The system program uses the status byte to store important information on the current status of the controller. The control byte can be used for making important requests via the STEP 5 program.

H Flag Doubleword

The user specifies the number of the 'H flag doubleword' during his configuring session with COM 155H. Its contents are of no consequence. If no number is specified for the flag doubleword, each entry in the error data block is automatically stamped with the current date and time from the internal CPU clock.

Each time it makes an entry in the error data block, the 155H system program copies the contents of the flag doubleword to the 6th and 7th data word of each error record.

Intermittent Outputs

In contrast to a non-intermittent output, an output is said to be intermittent when it changes its signal state at least once every hour.

Locating Facility (LF)

The 'locating facility' locates errors on redundant digital input/output modules.

For each redundant digital input or output for which the 155H system program is not only to detect but also to locate errors, the user must configure one additional input and one additional output.

NON-STOP Operation

The S5-155H tolerates the first failure of each redundant hardware component. The defective components can thus be repaired or replaced without having to interrupt program processing.

Passivation

When a redundant I/O module or communications processor (CP) fails, the 155H system program, after locating the problem, shuts it down (that is, passivates it) and operation is one-sided until depassivation takes place.

When an I/O module has to be passivated, the associated group power supply sometimes has to be shut down, thus passivating all other redundant modules that share the same power supply.

Once the module goes back on line, it must be depassivated!

Depassivation is usually done in the Restart routine in response to a user request. The relevant bit is set in the H flag word (acknowledgement key).

Readback Delay

The digital readback inputs (R-DIs) detect errors in redundant digital outputs. The 'readback delay' is the amount of time by which the digital readback inputs (R-DIs) should be delayed.

This delay makes it possible to take into account the different signal propagation times of the various digital output modules. The default readback delay is 0.01 s, but the user may define a delay of from 0.01 to 1.0 s with COM 155H. If the specified readback delay is shorter than the scan cycle time, the 155H system program takes the scan time as readback delay time.

Redundant Mode

When the S5-155H operates in Redundant mode, the master controls the process. The standby is updated at each synchronization point and checks to make sure that the master is still functioning properly. If the standby controller detects a fault in the master, it simply takes over control.

Redundant I/Os

An I/O module is 'redundant' when it is 'double', i.e. when it has the same address in both subunits. If one of the two I/O modules or subunits fails, the failure is tolerated and their is no interruption in the process. The defective module is reported and can then be replaced or repaired.

Self-Test

The S5-155H supports NON-STOP operation of redundant components with a number of comprehensive self-tests. These check the contents and state of the CPUs and I/Os and make comparisons between the two subunits. The tests execute in the background, and are transparent to the other software, until a hardware failure is detected and located. Every problem detected by the self-tests is reported.

Standby Activation

When the master controller is in Cyclic mode and the standby controller is to be (re)-instated in the process, for instance during a restart or error search, the 155H system program provides the standby with all of the data from the master. If necessary, it even transfers the whole user program from the master to the standby controller (as long as the program is in RAM).

The transfer of this static data can take **several cycles**. The standby controller is then updated; that is, within one cycle, it is given the master's dynamic data (see 'Updating'). The standby activation phase is complete when the internal states of master and standby are identical.

Standby-Master Switchover

Certain events necessitate making the standby the new master, and 'demoting' the master to standby status.

In Redundant mode, for instance, when the standby detects a fault in the master controller, the standby takes control as soon as the activation phase has been completed.

The event-driven synchronization of the two subunits ensures that bumpless switchover is possible at any time.

Solo Mode

In 'Solo mode', the master controls the process alone. The standby is at STOP or in Error Search mode, and does not participate in the process.

In Solo mode, the S5-155H is like an S5-155U; that is, the two subunits are not synchronized, but the master continues to execute the self-tests on its own.

Stuck at 0 and Stuck at 1 Errors

A certain input or output shows stuck at 0 or stuck at 1, and can no longer reacts to a signal change. The 155H system program continually checks all redundant input and outputs for stuck at 0 or stuck at 1.

Synchronization Points

Whever an event occurs which can result in the master and the standby having different internal states, the subunits are synchronized. They are, for instance, synchronized after every direct access operation, timer scan, and a process or timed interrupt. In the case of interrupts, the 'synchronization point' is always the next block change.

The 155H system program monitors each synchronization point. A check is made to make sure that both subunits are still working and that they are both processing the same operation.

Test Slices

The S5-155H's self-test comprises approximately 300 'test slices' of 2 ms duration each. The user can specify the exact number of test slices to be processed per cycle with COM 155H. The default value is one test slice per cycle, but the user may select a number between 1 and 19.

Transfer Data

'Transfer data' are the DB and DX data blocks whose contents are modified during processing of the STEP 5 program. The 155H system program must transfer all of these blocks from the master to the standby every time the standby controller is updated in order to ensure that the two subunits always have the same data.

The numbers of the DB and DX data blocks to be transferred must be specified in a COM 155H session at the programmer.

Updating

Updating is understood to be the procedure, undertaken by the 155H system program, of copying the dynamic data from the master controller, which is in RUN mode, to the standby controller **within one cycle**.

Updating is part of "standby activation", and is regarded as completed when the internal states of master and standby are identical.

SIEMENS

SIMATIC S5

Data Handling Blocks for the CPU 948R (Standard Function Blocks)

Reference Manual (S5-155H, Part II)

C79000-B8576-C134-02

Preface (How to Use This Manual)

Using the Data Handling Blocks	1
Assigning Parameters to the Data Handling Blocks	2
Data Handling Blocks in the User Program	3
Description: Data Handling Blocks	4

Copyright © Siemens AG 1994 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.

Technical data subject to change.

Siemens AG
Automation Group
EWK Elektronikwerk Karlsruhe

6ES5 998-4SR21

Preface (How to Use This Manual)

In SIMATIC S5 programmable controllers, specific tasks are handled by special modules such as communications processors and intelligent I/O modules. Data exchange between these modules and the CPUs is implemented by data handling blocks (DHBs).

This part contains a general introduction to, and instructions for, assigning parameters to the data handling blocks (FBs), followed by a description of how to evaluate the output parameters and detailed function descriptions of the data handling blocks for the CPU 948R (S5-155H).

Target Group

This Reference Manual is aimed at programmers with special system knowledge. If you have any questions which are not answered by the Reference Manual, please contact your local Siemens representative.

Notes on the Contents

The information below concerning the contents of the individual chapters is designed to make using this Reference Manual easier.

Chapter 1: Using the Data Handling Blocks

This chapter gives you basic information concerning the functions, possible applications and principle of operation of the CPU 948R data handling blocks. The introduction is followed by a description of the program sequence with data exchange via the data handling blocks.

Chapter 2: Assigning Parameters to the Data Handling Blocks

This chapter contains general statements concerning parameter assignment of the data handling blocks, the meaning of the parameters, methods of parameter passing and the types of parameter assignment with examples of direct and indirect parameter assignment.

The structure of the function block parameters is explained, especially the source and destination parameters, with reference to the different types of parameter assignment.

Chapter 3: Data Handling Blocks in the User Program

This chapter contains important notes for working with data handling blocks in the user program and for using data blocks in different functions.

It tells you how to calculate the available area length remaining for data transfer and also offers information on the execution time of the data handling blocks.

Chapter 4: Description: Data Handling Blocks

This chapter shows every available handling block with block diagram, parameter table and detailed functional description.

Index

The alphabetical index at the end of the manual will help you locate the most important terms in the manual.

Remarks Form

The (green) remarks form at the end of the manual is provided for your comments and recommendations.

Reference Literature

See the Preface to "S5-155H Programmable Controller" (Part I) in this manual.

Contents

1 Using the Data Handling Blocks II/1-1

1.1 Introduction II/1-2
Configuration II/1-2
Function Blocks II/1-4
Data Handling Blocks, Functions II/1-4
Addressing the Dual-Port RAMs II/1-4

1.2 Program Sequence II/1-5
Data Exchange II/1-5
Initializing an Interface II/1-5
Cold Restart, Cold Restart with Memory II/1-5

2 Assigning Parameters to the Data Handling Blocks II/2-1

2.1 Parameters and Function of the Data Handling Blocks II/2-2
Parameters and Parameter Passing II/2-2

2.2 Direct and Indirect Parameter Assignment II/2-3
Parameter Assignment Types II/2-3
Assigning Parameters to the SSNR, A-NR, ANZW and BLGR II/2-3
Examples of Assigning Parameters to the SSNR, A-NR and ANZW II/2-4
Assigning Parameters to Q/ZTYP, DBNR, Q/ZANF and Q/ZLAE II/2-5
Examples of DB Structure in the Case of Indirect Parameter Assignment II/2-6

2.3 Parameter Description II/2-7

2.4 Source and Destination Parameters
(Q/ZTYP, DBNR, Q/ZANF and Q/ZLAE) II/2-10

2.5 Evaluating the Output Parameters II/2-16
Parameters: PAFE and RLO II/2-16
Parameter: ANZW II/2-19
Writing to Condition Codewords in the Case of SEND/RECEIVE ALL ... II/2-20
Job Number II/2-20
Quantity of Transmitted Data II/2-21
Job Status II/2-22
Structure of the Condition Codeword II/2-23

3 Data Handling Blocks in the User Program II/3-1

3.1 Calling Data Handling Blocks II/3-2

3.2 Calling SEND-ALL and RECEIVE-ALL II/3-3

3.3 Using Data Blocks II/3-4

3.4 Memory Organization, Area Limits II/3-5

3.5 Runtime II/3-8

4 Description: Data Handling Blocks II/4-1

4.1 SEND Function Block (FB 120) II/4-2

4.2 SEND-A Function Block (FB 126) II/4-5

4.3 RECEIVE Function Block (FB 121) II/4-6

4.4 REC-A Function Block (FB 127) II/4-9

4.5 FETCH Function Block (FB 122) II/4-10

4.6 CONTROL Function Block (FB 123) II/4-12

4.7 RESET Function Block (FB 124) II/4-13

4.8 SYNCHRON Function Block (FB 125) II/4-14

Using the Data Handling Blocks

1

This chapter contains basic information concerning the functions, possible applications and principle of operation of the data handling blocks.

1.1 Introduction

SIMATIC S5 Communications

With the SIMATIC S5 programmable controller, tasks such as operator interfacing or bus links are supported by specific modules. The CP 143, for instance, is used for linking programmable controllers to the SINEC H1 bus-type local area network (LAN). In addition to such communications processors (CPs), intelligent I/O modules (IPs) are also used, for example, for closed-loop control and positioning.

The data handling blocks enable CPU ↔ CP and CPU ↔ IP communications (see Figure 1-1), but not between the CP and/or IP. These are function blocks which can be called up and assigned their parameters by the user.

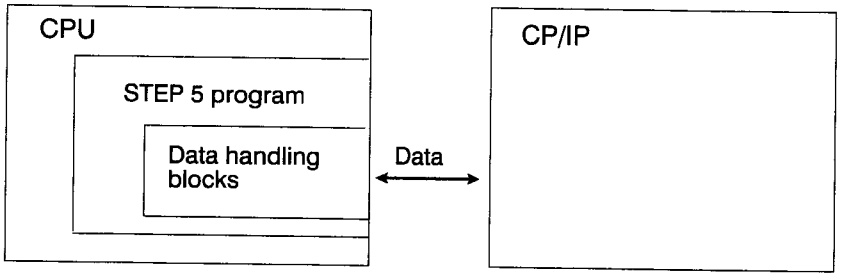


Figure 1-1 Data Communication between the CPU and the Intelligent I/Os

For reasons of simplicity, "CPU" will often be used below to refer to the CPU 948R and "CP" to refer to CPs and IPs using data handling blocks.

Configuration

The CPU accesses I/O modules via the S5 bus (read inputs, write outputs), using certain STEP 5 statements or instructions (L PW, T PW). Data exchange between CPU and CPs is carried out in a similar way.

The most important component of a CP module is a special memory (dual-port RAM, of which there may be several). The CPs themselves do not use the S5 bus, but fetch from the dual-port RAMs information stored there by the data handling blocks, or they store information there which is read by the data handling blocks.

The dual-port RAM itself and the hardware and/or software facilities which write data to the dual-port RAM and read data from the dual-port RAM and which establish the link with the actual CP (function) will be referred to below as the "interface". An interface is selected with the "SSNR" parameter (interface number, see parameter description).

The CP function could, for example, be the editing of data which it has received via the dual-port RAM or via the interface and transferring this data to a monitor or a bus. In this case, too, "interfaces" may possibly be used, but these should not be confused with those mentioned above!

Depending on their type, the CPs contain one or more dual-port RAMs or interfaces.

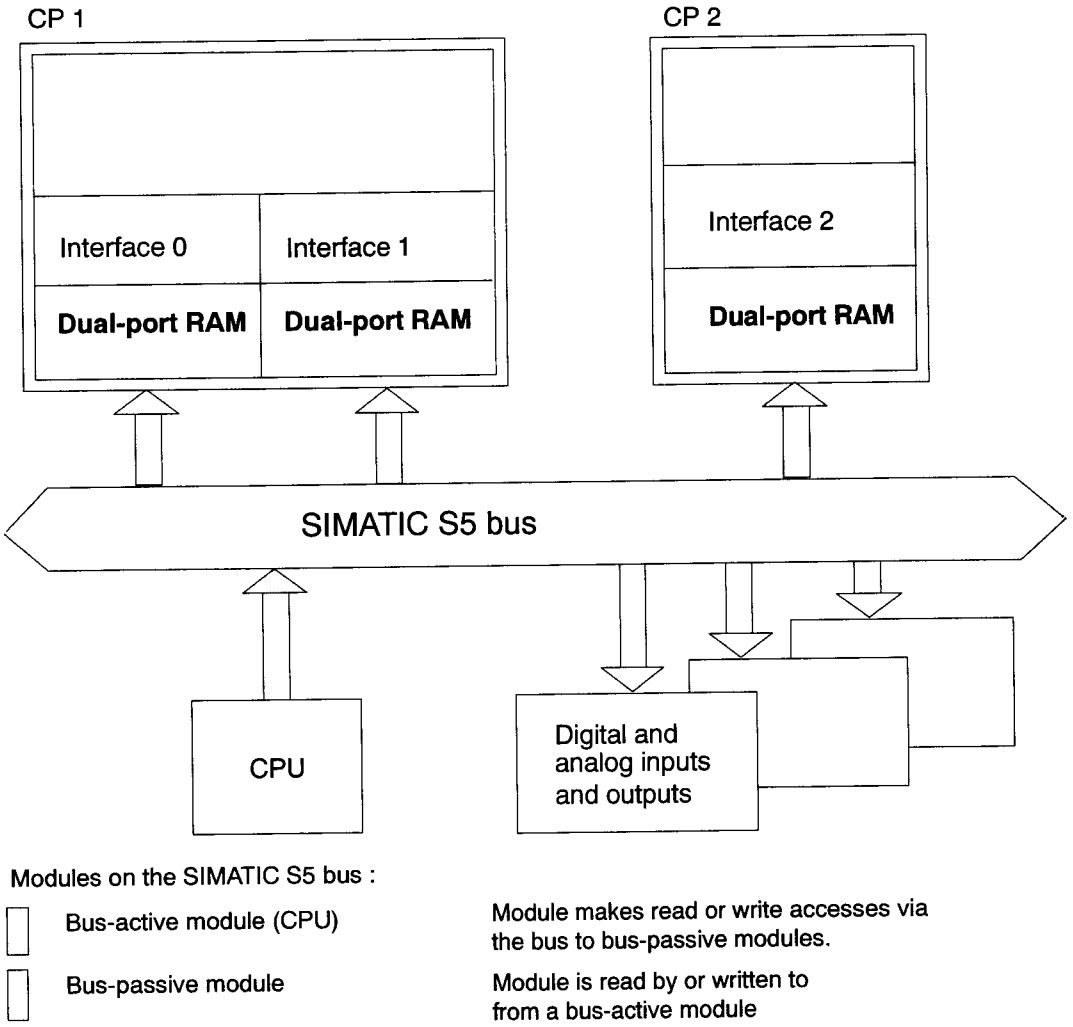


Figure 1-2 Basic Principle of a Configuration with Three CPs with a Total of Two Interfaces

1

The "DHBs for CPU 948R"

All CP/IP operating modes, one-sided and switched, are possible using the programming package "Data Handling Blocks for the CPU 948R".

The data handling blocks for the S5-155H are included in the scope of supply of the COM 155H programmer software and are located on diskette in the program file S5CR70ST.S5D. The data handling blocks are called up and the condition codes are evaluated in the same way as for the S5-155U.

Special features of the 155H DHBs compared to the 155U DHBs

- The CPs can be synchronized both at startup (OB 20, 21, 22) **and also in the cycle** (see: Calling the DHB SYNCHRON in the cycle).
- When assigning the source and destination type parameters, the specification "PY" (I/O (peripheral) byte) is permissible **only** for switched I/O. Both the CPs/IPs and the I/Os are then operated in switched mode.

Function Blocks

The following data handling blocks are available to the CPU 948R for the purpose of transferring data, parameters and control/status information from the CP/IP (more precisely, from the dual-port RAM) and to the CP/IP:

Function Block	Name	Function	Length
FB 120	SEND	Send data	approx. 3200 words 1)
FB 121	RECEIVE	Receive data	approx. 3200 words
FB 122	FETCH	Fetch job	approx. 1600 words
FB 123	CONTROL	Status inquiry	approx. 800 words
FB 124	RESET	Reset	approx. 600 words
FB 125	SYNCHRON	Initialize	approx. 600 words
FB 126	SEND-A	Send data	approx. 2400 words
FB 127	REC-A	Receive data	approx. 2400 words

[1] 1 word = 2 bytes = 16 bits

Note:
It is possible to change the function block numbers (but not those of the FX function blocks).

Data Handling Blocks, Functions

The data handling blocks are used in the extended area of the system data (RT area) and do not call data blocks.

The handling blocks access flags or data words or other areas which are used as parameters or contain parameters or the data to be transmitted or received.

In the same way, the handling blocks influence the result bits (CC 1, RLO etc.). Once processing of a handling block is completed, all result bits except RLO (result of logic operation) are irrelevant. RLO is designed as an error bit and can be set (error) or cleared (no error).

The accumulator contents (ACCU 1, ACCU 2, etc.) can be changed by means of the handling blocks.

Addressing the Dual-Port RAMs

The information in this section will be required for the jumper setting on the CP modules. Individual CP descriptions are contained in the relevant chapters:

Addressing procedure for data handling blocks:

"Page frame" addressing; the page number is identical to the parameter "SSNR" or interface number (see Section 2.3 Parameter Description).

1.2 Program Sequence

Data Exchange

A special coordination procedure enables different types of data/parameters to be transmitted in both directions via an interface (a dual-port RAM). This procedure is known as "handshaking".

The SEND, SEND-A, RECEIVE, REC-A, FETCH and RESET blocks only carry out handshaking if the control/status information read in previously requires and permits it. Otherwise, if there is no handshaking, the block is "idling".

The CONTROL block is restricted to the reading of status information. It does not carry out a handshake.

SYNCHRON block: see below.

Initializing an Interface

Provision must be made for interlocking the SYNCHRON DHB (FB 125) when initializing, either in one-sided or switched I/O, depending on the type of application of the CP.

Every interface must first be initialized by means of a SYNCHRON. This includes erasing/pre-programming of the dual-port RAM. This data handling block can be called up during restart; that is, during a cold restart or cold restart with memory (restart OBs: OB 20, OB 21, OB 22), and also during restart from "soft STOP" (OB 38) and in the cycle.

The remaining SEND, SEND-A, RECEIVE, REC-A, FETCH, CONTROL and RESET handling blocks can only communicate properly with an interface if this interface has previously been initialized without error.

Cold Restart, Cold Restart with Memory

If an interrupt has occurred when processing a data handling block, the handling block will **not** be continued at the interrupt point in the case of a cold restart with memory. Instead, OB 1 is processed from the beginning. This means:

Cold restart with memory of the CPU can be used; the CPs must be re-initialized.

Passivation of CP

In the event of an error, the CP affected is passivated. It then no longer carries out data transfer.

Depassivation of the interfaces is handled as usual via the H flag control byte.

Assigning Parameters to the Data Handling Blocks

2

This chapter contains statements concerning the parameter assignment of the data handling blocks, the meaning of the parameters, the methods of parameter transfer and the types of parameter assignment with examples of direct and indirect parameter assignment.

The structure of the function block parameters is described, especially the source and destination parameters, with reference to the different types of parameter assignment. The error bits which occur in the event of errors in processing the handling blocks and the meaning of the bit values are explained.

2.1 Parameters and Function of the Data Handling Blocks

Parameters and Parameter Passing

Parameters are assigned to all the handling blocks by a uniform procedure. The function block parameters will therefore be described in detail in this chapter. These are the parameters used:

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number
ANZW	I	W	Condition codeword
BLGR	D	KY	Frame size
QTYP/ZTYP	D	KS	Type of data source/data destination
DBNR	D	KY	Data block number: DB no. or DX no. or number of the address area (AS operations)
QANF/ZANF	D	KF	Relative start address within the type of data
QLAE/ZLAE	D	KF	Number (length) of the source/destination data
PAFE	Q	BY	Byte for parameter assignment errors
RLO			Result of logic operation, RLO bit, is used as an additional parameter

Each data handling block carries out a specific function. The SEND block, for example, is designed for data transfer from a CPU to a CP. Parameters determine which data is to be transferred. There are different ways of passing parameters to data handling blocks:

- Parameters at the function block
  The actual parameters specified with the function block call are used immediately. This procedure is called **direct parameter assignment.**
- Parameters in the data block
  The actual parameters of the function block call point to data words in a DB/DX data block. The data handling block uses the parameters contained in these data words. This procedure is called **indirect parameter assignment**.
- Parameters from the CP
  The CP provides the data handling block with the parameters required. The actual parameters of the function block call are irrelevant.

Depending on the method of passing the parameters and on the function, only some of the actual parameters may be required at the function block. In this case, values without any relevant meaning can be assigned as dummy parameters. It is also possible to use a block call with a reduced number of parameters (see SEND-A, REC-A).

Rule:
Parameters and/or flag words/data words which are completely irrelevant may contain any values. Parameters and/or flag words/data words which only contain the information of one byte must have "0" in the high-order byte.

2.2 Direct and Indirect Parameter Assignment

Parameter Assignment Types

As the table below shows, assignment of the function block parameters to the data handling blocks is clearly divided into three sections (parameter assignment groups).

FB Parameter	Data Handling Block									
	SEND	SEND-A	RECEIVE	REC-A	FETCH	CONTROL	RESET	SYNCHRON		
SSNR	x	x	x	x	x	x	x	x	SSNR	Section 1
A-NR	x	x	x	x	x	x	x	x	BLGR	
ANZW	x	x	x	x	x	x			ANZW	
QTYP	x		x		x				ZTYP	Section 2
DBNR	x		x		x				DBNR	
QANF	x		x		x				ZANF	
ALAE	x		x		x				ZLAE	
PAFE	x	x	x	x	x	x	x	x	PAFE	Section 3

The parameters of section 1 can either be specified directly or indirectly (the procedure, once selected, has to apply to all the parameters of the section).

Independent of section 1, the parameters of section 2 can also be specified directly or indirectly (same restriction as above) or are passed to the handling block by the CP.

Only the PAFE byte (chapter 3) is always directly specified.

With direct parameter assignment, the data handling block immediately processes the parameters specified in the block call. With indirect parameter assignment, a pointer to a parameter field is passed with the block parameters to the handling block. The "actual" parameters are located here in an unbroken sequence, and the order is the same as with direct parameter assignment.

Assigning Parameters to the SSNR, A-NR, ANZW and BLGR

The high-order byte of the SSNR parameter is used as the changeover criterion for direct/indirect parameter assignment (see example).

- **SSNR high-order byte = 0: direct parameter assignment**

  SSNR, A-NR and ANZW or BLGR (if the data handling block knows both these parameters) are actual parameters of the function block.

- **SSNR high-order byte ≠ 0: indirect parameter assignment**

  SSNR, A-NR and ANZW or BLGR (if the data handling block knows both these parameters) are stored in the selected data block (DB or DX) starting at the data word specified by the low-order byte of the actual parameter of SSNR.

SSNR and A-NR have the same data format (KY) in both types of parameter assignment. With the ANZW condition codeword, the formats differ. Whereas with direct parameter assignment, the address of the condition codeword is specified in STEP 5 notation (for example, FW 100, DW 17), with indirect parameter assignment, a doubleword is available. In the first data word, the area is in the KS data format (ASCII characters):

- FW stands for condition codeword in the flag area
- DB stands for condition codeword in the DB data block
- DX stands for condition codeword in the DX data block

In the second data word, the ANZW address is in the KY data format, and in the case of DB or DX, the block number is also included in the high-order byte.

Examples of Assigning Parameters to the SSNR, A-NR and ANZW

a) Direct assigning of parameters to the SSNR, A-NR and ANZW

```
STL                              Explanation
     .
     .
     JU       FB 120
     NAME  :  SEND
     SSNR  :  KY 0,21     SSNR parameter = 21
     A-NR  :  KY 0,33     A-NR parameter = 33
     ANZW  :  FW 100      ANZW = FW 100
```

b) Indirect assigning of parameters to the SSNR, A-NR and ANZW

```
STL                              Explanation
     .
+ ←C         DB 13       Calling a DB or DX data block
|    .
|    .
|    JU       FB 120
|    NAME  :  SEND        Interpreted as pointer
+ ←SSNR  :  KY 255,10    Irrelevant
|    A-NR  :  KY 0,0      Irrelevant
|    ANZW  :  FW 0
|    .
+ →DB        13          SSNR parameter = 21
+ →DW 10 :  KY 0,21     A-NR parameter = 33
     DW 11 :  KY 0,33     1st parameter for ANZW (area)
     DW 12 :  KS FW       2nd parameter for ANZW (address)
     DW 13 :  KY 0,100    ANZW = FW 100
```

Same example but with a condition codeword in the data block:

```
|    .
+ →DB        13
+ →DW 10 :  KY 0,21     SSNR parameter = 21
     DW 11 :  KY 0,33     A-NR parameter = 33
     DW 12 :  KS DB       1st parameter for ANZW (area)
     DW 13 :  KY 47,100   2nd parameter for ANZW (address)
                          ANZW = DW 100 in DB 47
```

c) Indirect assigning of parameters to the SSNR and BLGR

```
STL                              Explanation
     .
+ ←CX        DX 24               Calling a DB or DX data block
|    JU      FB 125
|    NAME :  SYNCHRON
+ ←SSNR :    KY 255,10           Pointer to parameter list
|    BLGR :  KY 0,0              Irrelevant
|    .
+ →DX        24
+ →DW 10:    KY 0,10             SSNR parameter = 10
     DW 11:  KY 0,5              BLGR parameter =  5
```

Assigning Parameters to Q/ZTYP, DBNR, Q/ZANF and Q/ZLAE

With **direct** parameter assignment, the data handling block directly processes the source parameters or destination parameters (consisting of QTYP/ZTYP, DBNR, QANF/ZANF and QLAE/ZLAE) specified in the block call.

With **indirect** parameter assignment, the block parameters form a reference to a parameter field in a data block which contains the "actual" source or destination parameters.

The indirect assignment of the source or destination parameters is carried out with the QTYP/ZTYP "XX" (KS data format). In the case of DBNR, the DB or DX block number must be specified in the low-order byte. If the high-order byte contains the value 0, the data handling block expects the parameter list in a DB data block, otherwise in a DX data block. QANF/ZANF contains the word number at which the parameter list begins.

QLAE/ZLAE is irrelevant.

In the case of operations with QTYP "AS" (absolute addressed memory locations), the required 64K memory area is selected via the DBNR parameter using the four highest-value address bits.

In this case, the QANF/ZANF parameter designates the low value (16-bit) part of the 20-bit address. (See Example c) and Section 3.4 "Memory Organization").

DB Structure in the Case of Indirect Parameter Assignment (TYPE = XX)

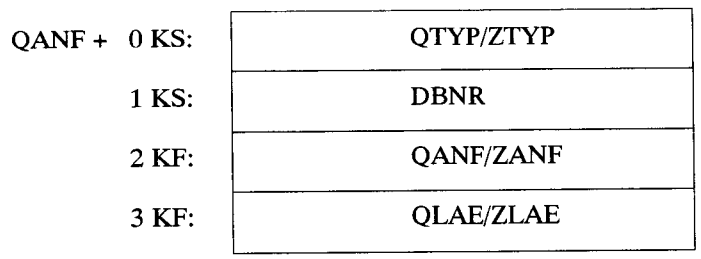


Figure 2-1 Data Block Structure in the Case of Indirect Parameter Assignment

Examples of DB Structure in the Case of Indirect Parameter Assignment

a) Direct assignment of parameters to QTYP, DBNR, QANF, QLAE

```
STL                                  Explanation
     .
     JU       FB 120
     NAME  :  SEND
     SSNR  :  KY 0,21
     A-NR  :  KY 0,33
     ANZW  :  FW 100
     QTYP  :  KS DB        QTYP parameter = DB
     DBNR  :  KY 0,17      DBNR parameter = 17
     QANF  :  KF 3         QANF parameter = from DW 3
     QLAE  :  KF 5         QLAE parameter = 5 words
     PAFE  :  QB 13
```

b) Indirect assignment of parameters to QTYP, DBNR, QANF, QLAE

```
STL                                  Explanation
     .
     .
     JU       FB 120
     NAME  :  SEND
     SSNR  :  KY 0,21
     A-NR  :  KY 0,33
     ANZW  :  FW 100
+ ←  QTYP  :  KS XX        Indirect parameter assignment
+ ←  DBNR     KY 0,25      Parameters are located in DB 25
+ ←  QANF  :  KF 11        from DW 11
|    QLAE  :  KY 0         Irrelevant
|    PAFE  :  QB 13
|    .
|
+ →  DB       25
+ →  DW 11:   KC DB        Parameter QTYP = DB
     DW 12:   KY 0,17      Parameter DBNR = 17
     DW 13:   KF 3         Parameter QANF = 3
     DW 13:   KF 5         Parameter QLAE = 5
```

c) Absolute addressing AS

```
STL                                          Explanation
Absolute address (hexadecimal)               1 AB00
-  comprises address of 64 K area            1
-  low-value part of the address               AB00
   (hexadecimal) or fixed-point                43776
```

```
     JU       FB 120
     NAME  :  SEND
     SSNR  :  KY 0,21
     A-NR  :  KY 0,33
     ANZW  :  FW 100
     QTYP  :  KS AS        AS operation
     DBNR     KY 0.1       64 K area: 1
     QANF  :  KF 43776     Low-value address 43776 = AB00H
     QLAE  :  KF 200
     PAFE  :  QB 13
```

2.3 Parameter Description

SSNR: Interface Number

The number of the interface at which the job to be addressed is located (see Section 1.1 Configuration).

- Parameter type/data type: Data/constant byte (D/KY; KY = 2 bytes)
- Permissible range: 0,0 ... 0,255 (direct parameter assignment)
  1,0 ... 255,255 –N (indirect parameter assignment)
  N = Number of words required for indirect parameter assignment
- High byte = 0: Direct parameter assignment; low-order byte = SSNR
- High byte ≠ 0: Indirect parameter assignment; low-order byte = pointerr

A-NR: Job Number

The job number addressed at the interface. Under the same number, a program or a parameter set, through which the function of the CP is defined, is located at the interface.

- Parameter type/data type: Data/constant byte (D/KY)
- Permissible range: 0,0 ("ALL" function)
  0,1 ... 0,223 ("DIRECT" function)

For the running of the blocks, there is an essential difference between the job number = 0 (an ALL function) and the job number ≠ 0. This operating mode is called DIRECT function (not to be confused with direct parameter assignment).

ANZW: Condition Codeword

Address of the condition codeword in which either the processing of the job specified under A-NR will be indicated to the user (DIRECT functions) or a job number (ALL functions).

- Parameter type/data type: Input/word (I/W)
- Permissible range: FW 0 ... 252
  DW 0 ... 254

With direct parameter assignment, DW refers to the DB/DX data block specified before the data handling block call. For indirect parameter assignment, see above.

The data blocks must be of sufficient length. If, for example, the ANZW address is DW 7, then the DB/DX block must be at least 14 words long (5 words for block header, DW 0 to DW 8), so that a possible subsequent ANZW (see below) can be processed.

The blocks which write to the condition codeword and the method of evaluating the latter are described in detail in Section 2.5 "Evaluating the Output Parameters".

BLGR: Frame Size

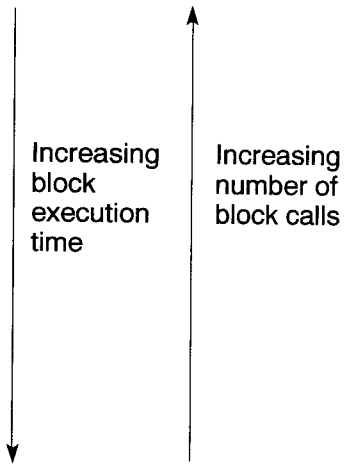
The maximum number of bytes which can be transferred by one SEND or RECEIVE block is called the frame size. If the quantity of data to be exchanged is greater than the frame size, as many blocks must be called as are necessary to complete the transfer.

On startup of the programmable controller, the frame size between the processor and the interface is decided by SYNCHRON. The frame size can be influenced by the BLGR parameter (see SYNCHRON block description).

- Parameter type/data type: Data/byte (D/KY)
- Permissible range : 0,0 ... 0,255

The following frame sizes can be set:

0	Frame with max. 256 bytes	1)
1	Frame with max. 16 bytes	
2	Frame with max. 32 bytes	
3	Frame with max. 64 bytes	
4	Frame with max. 128 bytes	
5	Frame with max. 256 bytes	
6	Frame with max. 512 bytes	
7..		
254	Frame with max. 256 bytes	2)
255	Frame fixed at 512 bytes	



1) Default for CPU 948; the block execution time is thus less than 10 ms (acknowledgement time not taken into account)

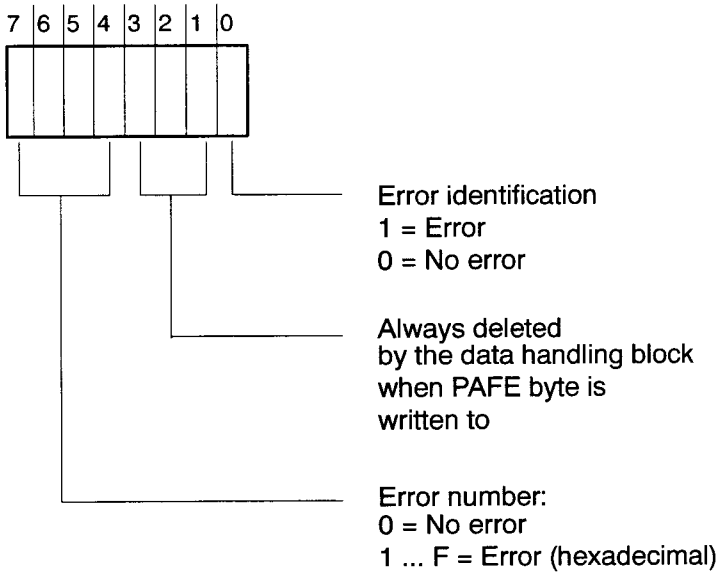
2) Corresponds to parameter assignment with frame size "0".

PAFE: Error Indication in the Event of Parameter Assignment Errors

The PAFE byte (flag byte, output byte or input byte) indicates when problems occur in processing a data handling block. When evaluating this parameter, indication of the (transitory or permanent) interface status (e.g. interface overloaded, not ready) must be distinguished from indication of "parameter assignment errors", for example, assignment of illegal parameters to A-NR, QTYP/ZTYP etc. In this case, the parameters assigned of the data handling block and/or the CP (interface) must be changed.

- Parameter type/data type: Output/byte (Q/BY)
- Permissible range: FY 0 ... FY 255
  QB 0 ... QB 127 (if present)
  IB 0 ... IB 127 (if present)

PAFE Byte Structure



The error numbers are described in detail in Section 2.5 "Evaluating the Output Parameters".

2.4 Source and Destination Parameters (Q/ZTYP, DBNR, Q/ZANF and Q/ZLAE)

Source Parameters: QTYP, QANF, QLAE

The term "source parameter" covers the following parameters:

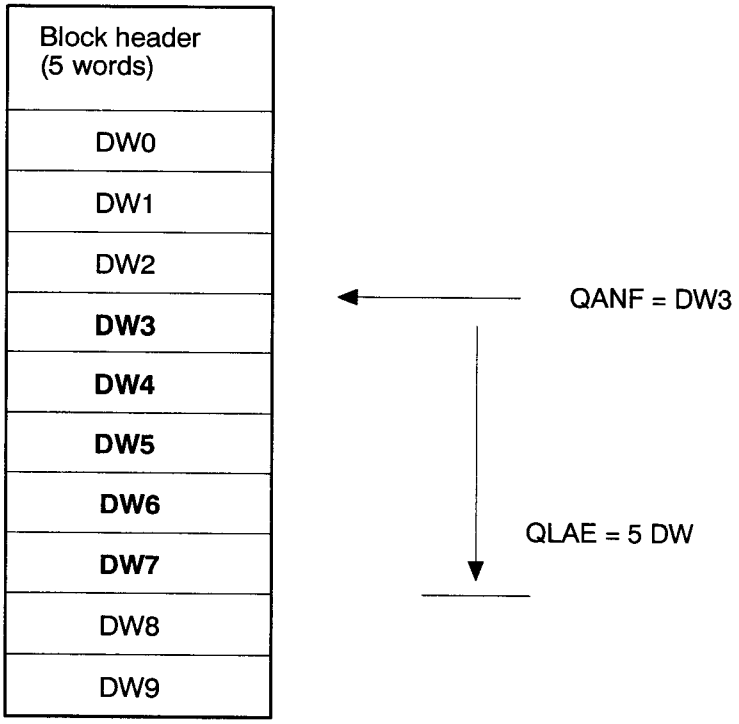
Table 2-1 Structure of the Source Parameter

Parameter	Meaning	Example
QTYP	Source type	DB
DBNR	Data block number or: 64K memory area in AS mode	(DB) 17
QANF	Source start	(DW) 3
QLAE	Source length	5 (DW)

These parameters identify an area. This area forms a data source; that is, the data of this area are transferred (copied) to the interface.

Example

QTYP/DBNR = DB17



Destination Parameters: ZTYP, ZANF, ZLAE

The term "destination parameter" covers the following parameters:

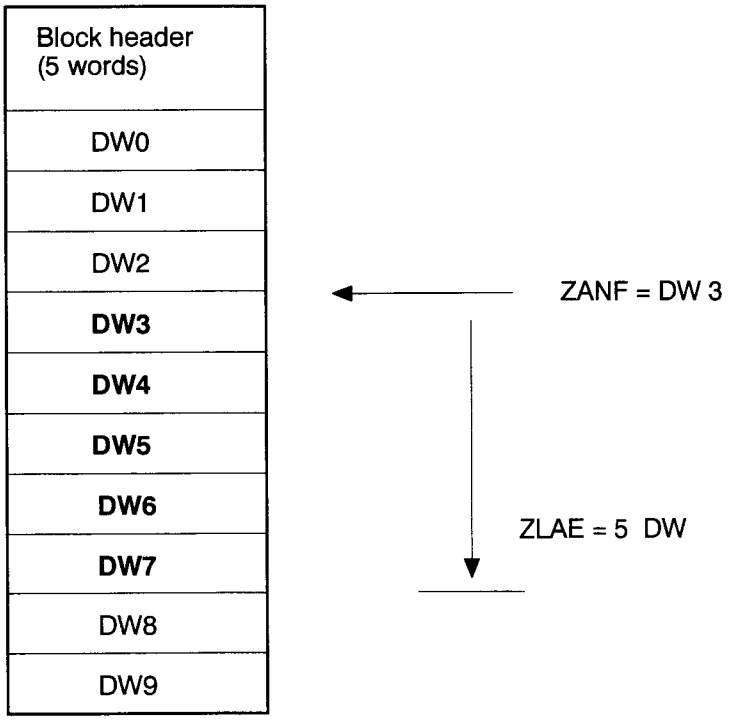
Table 2-2 Structure of the Destination Parameter

Parameter	Meaning	Example
ZTYP	Destination type	DX
DBNR	Data block number or: 64K memory area in AS mode	(DX) 18
ZANF	Destination start	(DW) 3
ZLAE	Destination length	5 (DW)

These four parameters identify an area. This area forms a data sink; that is, it receives data supplied from the interface. Do not forget that in this process the "old" data are overwritten.

Example

ZTYP/DBNR = DX 18



See Section 3.4 "Memory Organization, Area Limits" for further information.

2

QTYP/ZTYP: Type of Data Source/Data Destination

With this parameter, the type of data source (with SEND block) or type of data destination (with RECEIVE and FETCH blocks) can be specified by means of ASCII characters.

- Parameter type/data type: Data/constant character (D/KY; KS = 2 ASCII characters)
- Permissible area: DB, DX, ZB, TB, BS, AS (direct parameter assignment, word areas)
  MB, AB, EB, PB (direct parameter assignment, byte areas)
  XX (indirect parameter assignment)
  RW (READ/WRITE),
  NN (source/destination parameters of CP)

DBNR: DB Number with DB, DX, XX, RW Type Identifiers

If an XX, RW, DB or DX identifier is specified for QTYP/ZTYP, the data handling block must be informed of the data block required in the low-order byte of this parameter.

- Parameter type/data type: Data/constant byte (D/KY; KS = 2 bytes)
- Permissible area: 0,3 ... 0,255 (with direct parameter assignment, high-order byte must be 0)
  0,3... 255,255 (with indirect parameter assignment and READ/WRITE)
  if high-order byte = 0: DB data block
  if high-order byte $\neq$ 0: DX data block)

If, with QTYP, AS is assigned as the type of data source or data destination, the number of the 64K memory area should then be specified as a parameter for DBNR (see Section 3.4).

QANF/ZANF: Start Address of the Source/Destination Data Frame

Start address (relative to area start) of the source/destination frame with direct parameter assignment.

Using the XX (indirect parameter assignment) and RW (READ/WRITE) type identifiers, the DW numbers from which the parameters start can be specified here.

- Parameter type/data type: Data/constant fixed point (D/KF)
- Permissible area
  for CPU 948: 0 ... 65535

Note:
With the KF data format, the programmer accepts any value between –32768 and +32767. QANF/ZANF is considered by the handling block as an unsigned (positive) number between 0 and 65535 (corresponds to 0000H...FFFFH).

QLAE/ZLAE: Length of the Source/Destination Frame

Depending on the source/destination type, the information is understood to be in bytes or in words.

- Parameter type/data type: Data/constant fixed point (D/KF)
- Permissible area: 1 ... 32767, –1

The "joker length" (QLAE, ZLAE = –1) means:

- In the case of RECEIVE, that as much data is accepted as the interface provides
- In the case of SEND, that as much data is transferred to the interface as the area permits (transmission up to the limit of the area).

The meaning and relationships between the source/destination parameters are listed in the table below:

Table 2-3 Meaning and Relationships of the Source/Destination Parameters; Word Areas (1)

QTYP/ZTYP Description	**DB** Source/destination data from/to data block	**DX** Source/destination data from/to DX data block	**ZB** Source/destination data from/to counters
DBNR Meaning Permissible area CPU 948	DB from which the source data are taken or into which the destination data are transferred 3 ... 255	DX from which the source data are taken or to which the destination data are transferred 3 ... 255	Irrelevant
QANF/ZANF Meaning Permissible area CPU 948	DW number starting from which data are read or written 0 ... 4090	DW number starting from which data are read or written 0 ... 4090	Number of counter starting from which data are read or written 0 ... 255
QLAE/ZLAE Meaning Permissible area CPU 948	Length of the source/destination frame in words 1 ... 4091	Length of the source/destination frame in words 1 ... 4091	Length of the source/ destination frame in words (1 counter = 1 word) 1 ... 256

Table 2-4 Meaning and Relationships of the Source/Destination Parameters; Word Areas (2)

QTYP/ZTYP Description	**TB** Source/destination data from/to timers	**BS** Source/destination data from/to system data area	**AS** Source/destination data from/to absolute addressed memory locations
DBNR Meaning Permissible area CPU 948	Irrelevant –	Irrelevant –	64 K area address 0.0 ... 0.13
QANF/ZANF Meaning Permissible area CPU 948	Number of timer starting from which the data are read or written 0 ... 255	Number of RS word starting from which the data are read or written 0 ... 255	Absolute start address starting from which data are read or written 0 ... FFFF (hex.) or 0 ... 65535 (fixed-point dec.)
QLAE/ZLAE Meaning Permissible area CPU 948	Length of the source/ destination frame in words (1 timer = 1 word) 1 ... 256	Length of the source/ destination frame in words (1 system data item = 1 word) 1 ... 256	Length of the source/ destination frame in words (1 counter = 1 word) 1 ... 32767

See Section 3.4 "Area Limits" for further information.

Table 2-5 Meaning and Relationships of the Source/Destination Parameters; Word Area

QTYP/ZTYP Description	MB Source/destination data from/to flag area	AB Source/destination data from/to process image of the outputs (PIQ)	EB Source/destination data from/to process image of the inputs (PII)	PB Source/destination data from/to I/O modules, if source data: input modules, if destination data: output modules
DBNR	Irrelevant	Irrelevant	Irrelevant	Irrelevant
QANF/ZANF Meaning Permissible area CPU 948	Flag byte number starting from which data are read or written 0 ... 255	Output byte number starting from which data are read or written 0 ... 127	Input byte number starting from which data are read or written 0 ... 127	I/O byte number starting from which data are read or written. 0 ... 127 dig. I/O, 128 ... 255 dig. or analog I/O
QLAE/ZLAE Meaning Permissible area CPU 948	Length of the source/destination frame in bytes 1 ... 256	Length of the source/destination frame in bytes 1 ... 128	Length of the source/destination frame in bytes 1 ... 128	Length of the source/destination frame in bytes 1 ... 256

Table 2-6 Meaning and Relationships of the Source/Destination Parameters; Special Cases

QTYP/ZTYP Meaning	XX Indirect parameter assignment; Source or destination parameters are stored in the DB or DX data block (specified by DBNR and QANF)	RW READ/WRITE; Source or destination parameters are stored in the DB or DX data block (specified by DBNR and QANF)	NN No source/destination parameters in block; parameters can be supplied by interface
DBNR Meaning Permissible area CPU 948	DB/DX in which the source/destination parameters are stored; DB if high-order byte = 0 otherwise DX 1 ... 255	DB/DX in which the source/destination parameters are stored; DB if high-order byte = 0 otherwise DX 1 ... 255	Irrelevant
QANF/ZANF Meaning Permissible area CPU 948	DW number starting from which the parameters are stored 0 ... 32767	DW number starting from which the parameters are stored 0 ... 32767	Irrelevant
QLAE/ZLAE	Irrelevant	Irrelevant	Irrelevant

See Section 3.4 Area Limits for further information.

Data Block: Structure in the Case of Indirect Parameter Assignment

Type = XX

QANF +	0 KS	QTYP/ZTYP but not XX, RW, NN
	1 KY:	DBNR in the case of type DB, DX
	2 KF:	QANF/ZANF start address
	3 KF:	QLAE/ZLAE length

Figure 2-2 Data Block Structure in the Case of Indirect Parameter Assignment

Data Block: Structure in the Case of READ/WRITE

Type = RW

QANF +	0 KS	QTYP but not XX, RW, NN	"Data source" description
	1 KY:	DBNR in the case of QTYP DB, DX	
	2 KF:	QANF source start address	
	3 KF:	QLAE source length	
	4 KS:	ZTYP but not XX, RW, NN	"Data destination" description
	5 KY:	DBNR in the case of ZTYP DB, DX	
	6 KF:	ZANF destination start address	
	7 KF:	ZLAE destination length	

Figure 2-3 Data Block Structure in the Case of READ/WRITE

2

2.5 Evaluating the Output Parameters

Output Parameters: RLO, PAFE, ANZW

PAFE is purely an output parameter, whereas RLO and ANZW are both input and output parameters. The following graphical representation shows in what way and in what context the data handling blocks influence the output parameters.

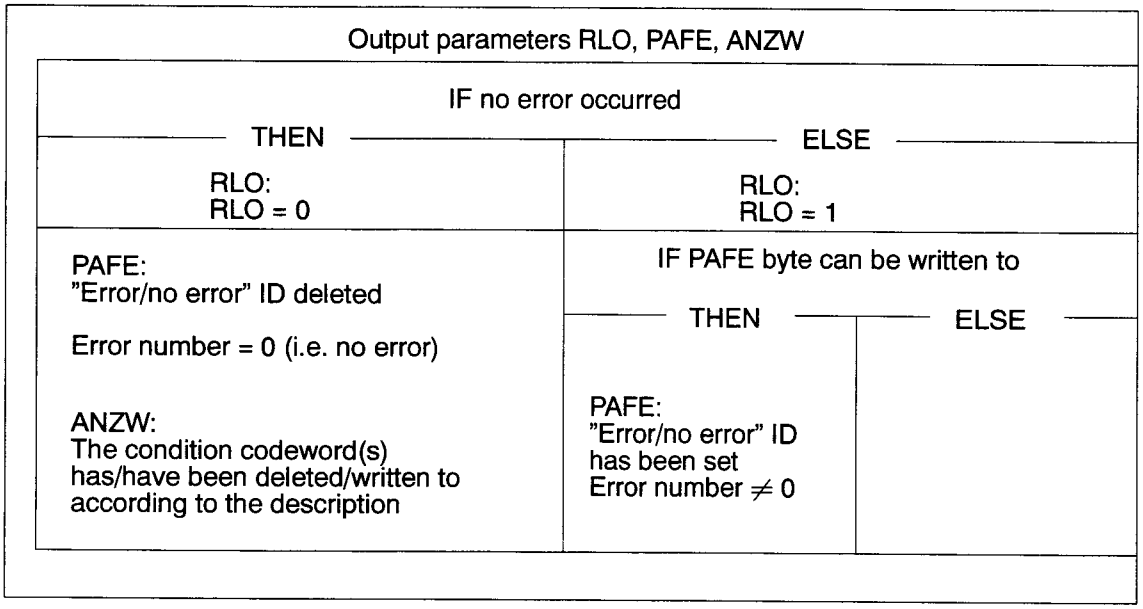


Figure 2-4 Influence of the Data Handling Blocks on the Output Parameters

If the condition accompanying "IF" is fulfilled, then the column under "THEN" is valid. If it is not fulfilled, the "ELSE" column is valid, etc.

If an error has occurred, the condition codeword is irrelevant.

Parameters: PAFE and RLO

If an error occurs during the data handling block routine (this can be an "actual" parameter assignment error, or other type of error [1]), the RLO is set. If no error occurs, RLO is reset (deleted). A quick error analysis can therefore be carried out following the function block [2].

Error indication by means of RLO always takes place. The PAFE byte can only then be written to if the PAFE (actual) parameter is permissible.

1) **Note:** A typical parameter assignment error is, for example, a job number greater than the permissible maximum. If the CP is not ready to communicate ("interface overloaded", "interface reserved by other processor"), this, as well as other errors connected with the CP, will be recognized as an error and indicated with an error number in the PAFE byte.

2) **Note:** No error has occurred if handshake signals are not exchanged (1-4) or if a handshake is broken off (5, 6) for any of the following reasons:

1. RLO (RLO "input parameter" not to be confused with RLO "output parameter" (= error indicator)) = 0
2. Job is already/still running (SEND/FETCH direct)
3. RECEIVE job is not (yet) ready (RECEIVE direct)
4. This function does not include a handshake (CONTROL)
5. Data transfer/reception is disabled
6. Only parameters are to be passed, etc.

It follows, therefore, that, in these cases too, no error numbers are defined in the PAFE byte. The individual block descriptions provide information as to which of these six situations may be applicable.

Structure: PAFE Byte

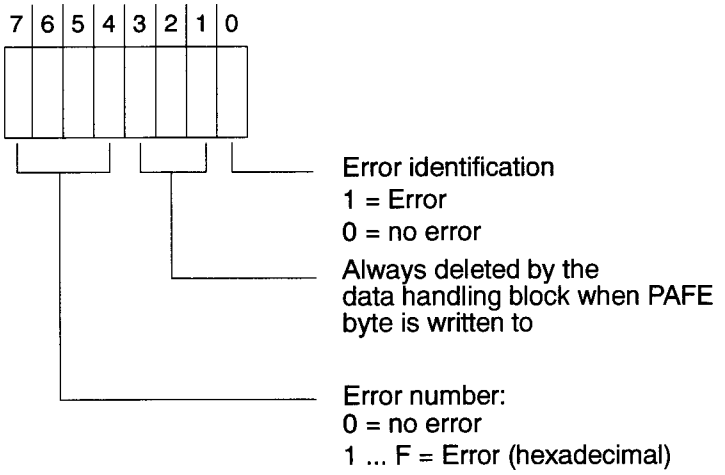


Table 2-7 Meaning of the Error Numbers in the PAFE Byte

Error No. (hex.)	Meaning
0	No error
1–F	Error
1–4	Error in source/destination parameter (QTYP/ZTYP, DBNR, QANF/ZANF or QLAE/ZLAE) or pointer to source/destination parameter incorrect with QTYP/ZTYP XX and RW, (XX = indirect parameter assignment, RW = READ/WRITE) or error in source/destination parameter from CP
1	Source/destination formally incorrect – QTYP/ZTYP illegal – DBNR data block number greater than 255; that is, high byte ≠ 0 – QLAE/ZLAE length illegal, permissible length: 1...32767 and –1.
2	DB or DX data block does not exist or is not permissible; (for example: DB0, DX0; in the case of QTYP/ZTYP DB, DX, XX, RW).

Error No. (hex.)	Meaning
3	Area too small or sum of start address (QANF/ZANF) and length (QLAE/ZLAE) too great (with all QTYP/ZTYP).
4	Area does not exist or is not permissible (with QTYP/ZTYP, AS, AB, EB, PB).
5	(Address of) condition codeword incorrect
6	No. not reserved
7	Interface does not exist
8	Interface not ready
9	Interface overloaded
A	Interface reserved by other processor (multiprocessor operation)
B	Job number illegal or frame size (SYNCHRON) illegal
C	Interface does not respond or interface does not respond in time or interface rejects job (negative handshaking acknowledgement).
D	Other interface errors, including – Error (or illegal acknowledgement) in handshaking acknowledgement – Frame size of interface not permissible – Synchronization running (no error)
E	Other data handling block, errors including – No data block selected with indirect assignment of parameters to the SSNR, A-NR, ANZW, BLGR – If PAFE is assigned EB or AB and does not exist
F	FB call not permissible, including – Double call when interrupts can occur between statements.

Note:
If the "Status" bit is set in the condition codeword ANZW and the identifier "Job completed with error" (bit $2^3 = 1$) is set, an error number has been written in the bits 2^8 to 2^{11}. This error number corresponds to that of the PAFE byte as follows:

1 ... 5	These error numbers and their meanings are identical in the PAFE byte and ANZW: The interface stores the number received from the data handling block in the job status without carrying out any change. After the corresponding parameters have been assigned, the job status is copied into the condition codeword by the next data handling block called.
6 ... F	The meaning of these numbers in the PAFE byte, as it is described here, is not identical to the numbering in the job statuses. The CP descriptions contain specific error lists.

Parameter: ANZW

The jobs identified by a job number constitute the management unit of an interface (or of the CPs).

The corresponding assignment on the handling block

- is indicated by the A-NR parameter (SEND/RECEIVE/FETCH/CONTROL/RESET direct)
- is produced by the flagging of a job number in the ANZW (SEND/RECEIVE/CONTROL ALL)

In the interface, there is a job status for every job. The status is managed by the interface and it indicates whether a job is (still) running in the CP or whether it has been completed by the CP without any errors or with a particular error. Occupation of the condition codeword with "status" (see below) means, among other things, that this job is copied into the ANZW.

The STEP 5 program in the CPU should be structured so that a particular condition codeword is allocated to each defined job.

In this way, an image of the activities in the interface or in the CP can be obtained in the CPU. This can only take place if the condition codewords are regularly updated by calling suitable data handling blocks (see block description, for example, CONTROL, SEND direct when idling).

You must distinguish between the ANZW 1 and ANZW 2 parameters.

Furthermore, do not forget that when ANZW 1 is occupied with "status", it will be used both as an input parameter (bit $2^7 = 1$: data transfer/reception disabled!) and as an output parameter.

Rules for Writing to ANZW by the DHB

1. The condition codeword(s) will basically only be written to (changed) if no error has occurred while the data handling block was being processed. In this case, the condition codewords will be written to according to the following table:

Table 2-8 Assignments of the Condition Codewords

Function	ANZW 1	ANZW 2
SEND/REC. DIRECT	Status	Quantity
SEND/REC. ALL	Job no.	–
FETCH (DIRECT)	Status	–
CONTROL DIRECT	Status	–
CONTROL ALL	Job no.	–

2. ANZW 1 and ANZW 2 (with SEND/REC DIRECT) is always written to (if no error has occurred).

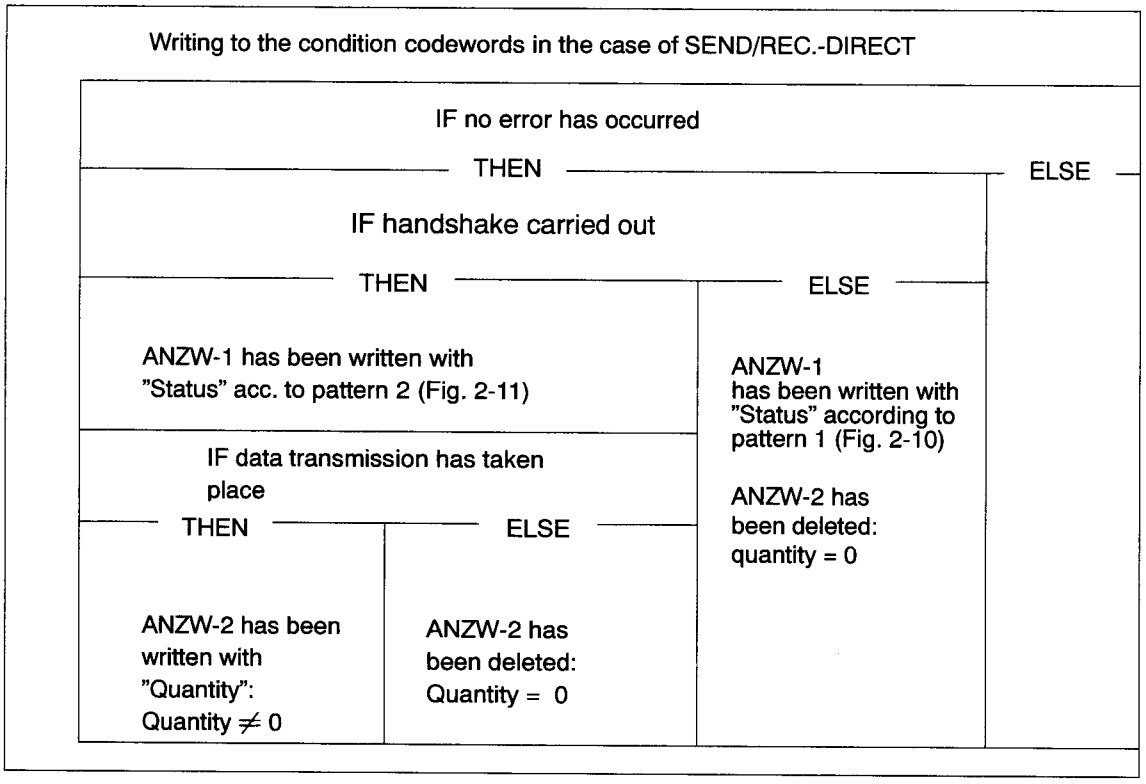


Figure 2-5 Influence of the Data Handling Blocks on the Output Parameters

Writing to Condition Codewords in the Case of SEND/ RECEIVE ALL

In the case of SEND/RECEIVE ALL, the job number is written to ANZW 1.

Job Number

If no error has occurred, the job number is written into the low-order byte and the high-order byte is deleted (ANZW 1).

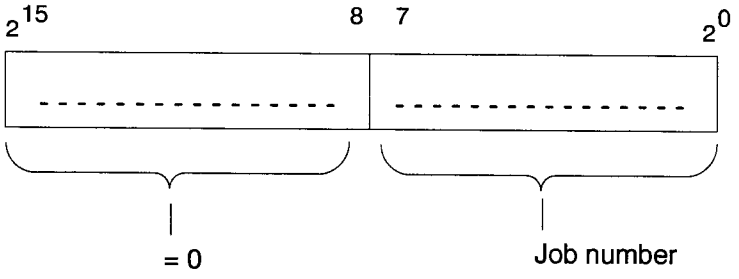


Figure 2-6 Job Number

CONTROL ALL

= 0: No job is being processed (by the interface/CP).

≠ 0: This job is being processed.

	SEND/RECEIVE ALL
= 0 :	Idling; that is, no handshake carried out.
1...223:	A handshake has been carried out for this job.
= 255:	A handshake has been carried out. The CP supplied an alternative number instead of the actual job number.

Quantity of Transmitted Data

If no data is transmitted and no error has occurred, the "Quantity" words (ANZW 2) are deleted. If data is transmitted, the quantity word contains the quantity of all the data already transmitted in this job which may be considerably more than the quantity of data transmitted by the last data handling block.

If the "Data transfer/data reception completed" bit (bits 5 and 6) in ANZW 1 is set, ANZW 2 contains the source length or destination length of the job (job number). This is always indicated in bytes.

ANZW 2:

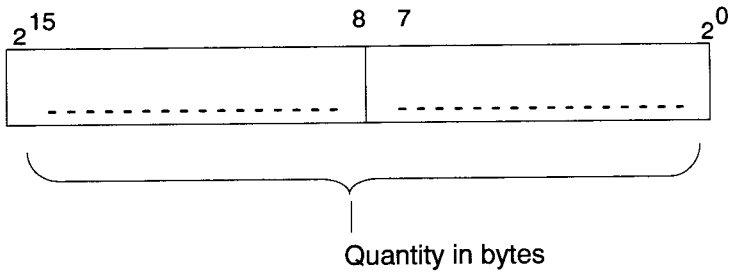


Figure 2-7 Quantity of Data to be Transferred

Job Status

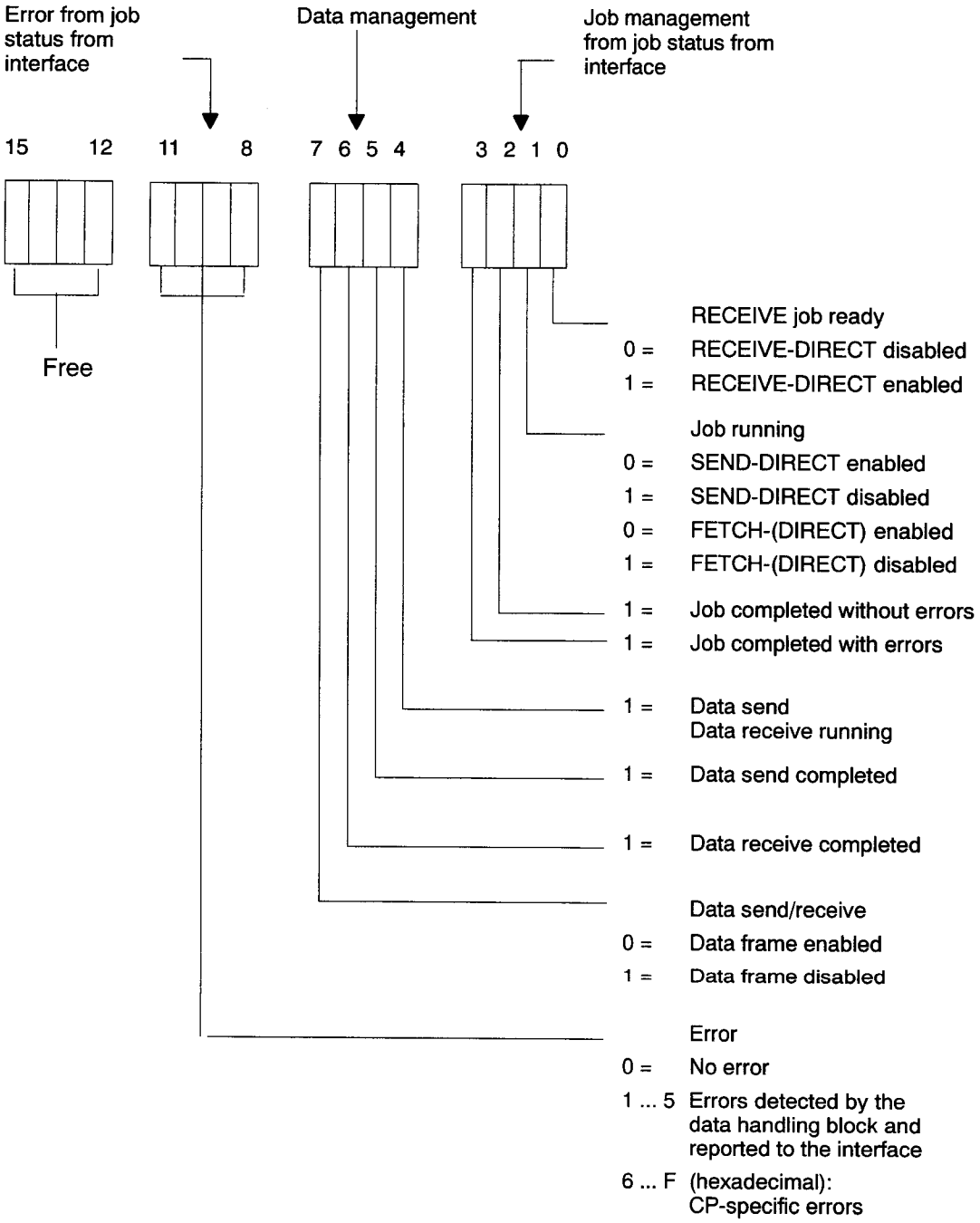


Figure 2-8 Condition Codeword

- Without handshake:

With SEND/REC./FETCH/CONTROL DIRECT, the job status must be read/evaluated by the data handling block before a handshake is carried out. This job status is written into ANZW 1 according to pattern 1 (Figure 2-9) if no handshake takes place (and no errors occur).

- With handshake:

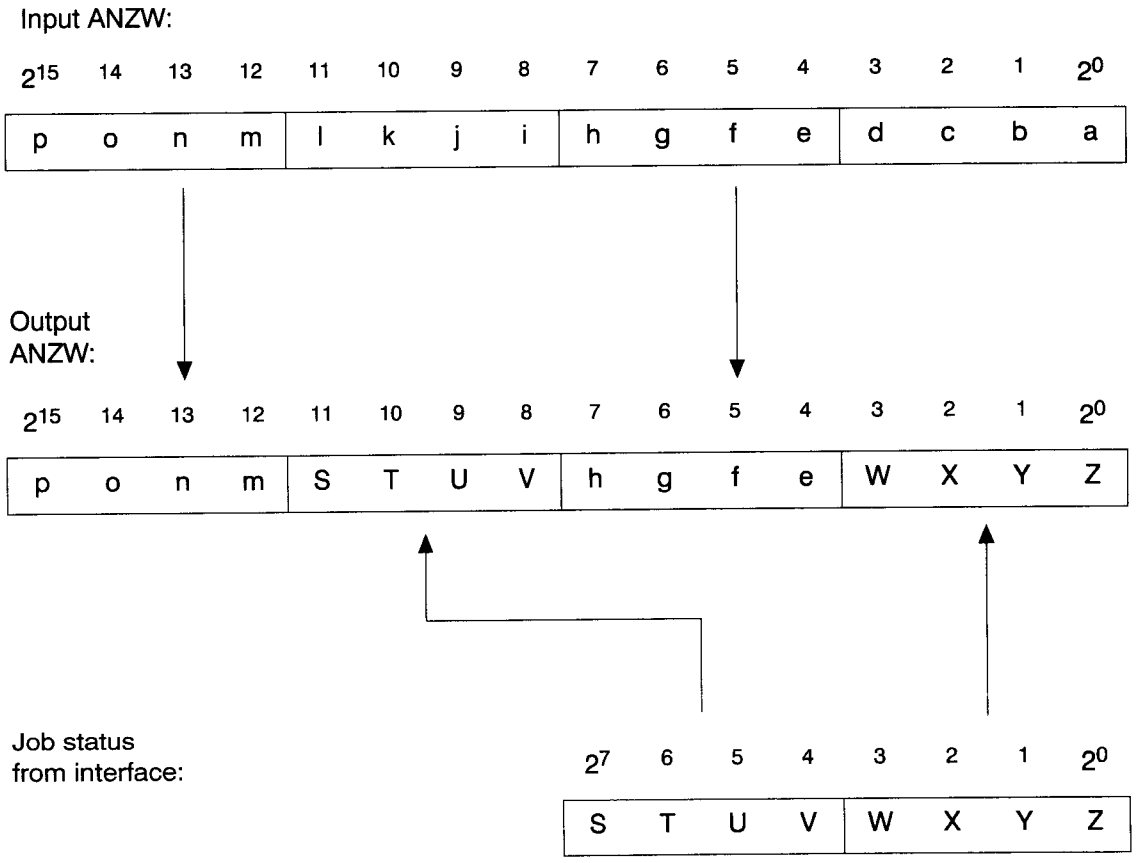
In the case of all functions where a handshake was initiated (and executed without error), the job status will be

- re-read (in the case of DIRECT functions) or
- read for the first time (in the case of ALL functions)

by the data handling block at the end of the handshake.

This updated job status does not have to be evaluated by the data handling block. It will be inserted in ANZW 1 according to pattern 2 (Figure 2-10).

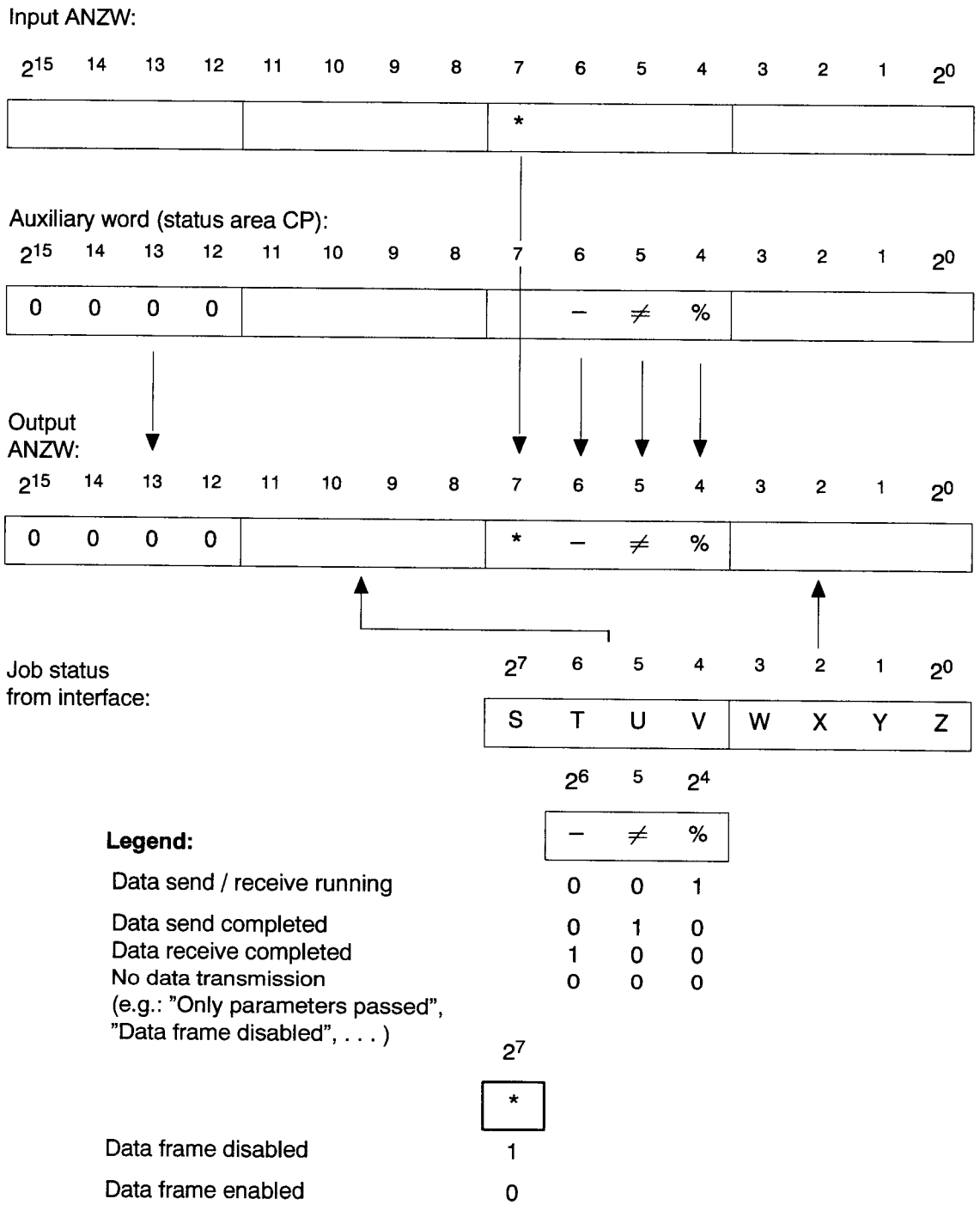
Structure of the Condition Codeword



Input ANZW: Condition codeword before data handling block is called

Output ANZW: Condition codeword after data handling block is called

Figure 2-9 Pattern 1



Input ANZW: Condition codeword before data handling block is called

Output ANZW: Condition codeword after data handling block is called

Figure 2-10 Pattern 2

Note:
If bit positions 2^{12} to 2^{15} have been deleted following the running of a data handling block, and if they were set (at least some of them) beforehand, then this is an obvious indication that:

1. The handling block was run without any errors (otherwise the condition codewords are not written to)
2. A handshake was carried out (otherwise the condition codeword would not be written to according to pattern 2).

Bit positions 2^4 to 2^6 now give information about whether data has been transmitted.

Division of the Condition Codeword

- **Nibble 1, bits 0 to 3, job management:**

The following is encoded here: whether a job has already been started or whether errors have occurred or whether the job has been disabled.

Important: Each of the bits has a significance of its own. In the case of several CPs, additional significance is assigned to the bit combinations. For example, if "Job running" (bit 2) and "Job completed" (bit 2 or 2) are set simultaneously (contradiction!) in the case of the CP 143, this has the supplementary meaning of "CP not ready for this job/job disabled".

- **Nibble 2, bits 4 to 7, data management:**

The following is encoded here: whether the data transfer for the job is still running or whether the transfer or reception of data has already been completed. With bit 7, the data block for the job can be disabled (bit 7 = 1: data block disabled; bit 7 = 0: data block enabled).

- **Nibble 3, bits 8 to 11, error number:**

These are the error indicators of the job. The error indicators are only valid if the bit "Job completed with error" in the first nibble is set at the same time.

- **Nibble 4, bits 12 to 15, free**

Operation of the "Status" Condition Codeword

a. **Bit 0** : RECEIVE job ready (handshake meaningful)

Setting/resetting:	By the data handling blocks corresponding to the deletion condition code in the job status. The bit "RECEIVE job ready" is used with the RECEIVE DIRECT function (received data available).
Evaluation:	By the RECEIVE block; the RECEIVE block carries out the handshake with the interface only if bit 0 is set to 1.
	– By the user to query whether or not received data is available.

b. **Bit 1** : Job running (SEND/FETCH disabled)

Setting/resetting: By the data handling blocks corresponding to the deletion condition code in the job status. The "Job running" bit (= 0) is used with the SEND DIRECT and FETCH DIRECT functions.

Evaluation: By the SEND block and FETCH block. The block will only carry out the handshake with the interface if bit 1 = 0; that is, a job will only be assigned if the "old" job has been processed.

– By the user in order to find out whether a "new" job can be initiated.

c. **Bit 2** : Job completed without errors

Setting/resetting: By the handling blocks corresponding to the deletion condition code in the job status.

Evaluation: By the user to check whether the job was completed by the interface without errors.

d. **Bit 3** : Job completed with error

Setting/resetting: By the handling blocks corresponding to the deletion condition code in the job status.

Evaluation: By the user to check whether the job was completed by the interface with an error. If the "Job completed with error" identifier is set, the reason for the error is to be found in bits 8 to 11 of the condition codeword.

e. **Bit 4** : Data transfer/reception running

Setting/resetting: By the SEND and RECEIVE handling blocks. If the transfer/reception for a job was started (1st subframe transferred) and not yet completed; for example, if data is still to be exchanged via the ALL functions. For this purpose, the initiation and transmission of the first subframe must have been carried out with a DIRECT function.

Deleting: By the handling blocks if the condition for setting has not been fulfilled.

Evaluation: By the user.
During CP-CPU data transmission, you may no longer change (SEND) or use (RECEIVE) the data frame of a job.
With "small" data frames, this is not critical since data transmission here requires only one block call and leads immediately to the condition code "Data transfer completed" or "Data reception completed". Larger data frames can, however, only be transmitted in subframes, and this requires several data handling block calls. In order to maintain the integrity of the data, you must first check whether the data frame has been transmitted completely before you change/use the data of a job. Otherwise, "old" and "new" data would be mixed.

f. **Bit 5** : Data transfer completed

Setting:	By the SEND handling block if the data of a job has been transferred complete to the interface (last subframe transmitted).
Resetting:	By the handling blocks if the setting condition has not been fulfilled. By the user if evaluation has taken place (edge generation).
Evaluation:	By the user With this bit, you can ascertain whether the data for a job has already been transferred complete to the interface.

g. **Bit 6** : Data reception completed

Setting:	By the RECEIVE handling block if the data for a job has been received complete by the interface (last subframe transmitted).
Resetting:	By the handling blocks if the condition for setting has not been fulfilled. By the user if evaluation has taken place (edge generation).
Evaluation:	By the user. With this bit, you can ascertain whether the data of a job has already been transmitted to the CPU.

h. **Bit 7** : Data transfer/reception disabled

Setting:	By the user in order to prevent a data frame being written to by the RECEIVE block or being read out by the SEND block.
Resetting:	By the user in order to enable the relevant data frame.
Evaluation:	By the SEND and RECEIVE handling blocks once only before the first subframe. If bit 2^7 is set, the blocks do not transfer any data. Instead, they report the "error" to the interface.

i. **Bit 8 to 11** : Error number

Setting/resetting:	By the handling blocks according to the indicator in the job status.
Evaluation:	By the user.

The following indicators can appear:

0:	No error
1 to 5:	CPU or handling block error The meanings of the error numbers one to five are identical to the numbers in the PAFE byte (parameter assignment error).
6 to F :	CP or interface error The error numbers 6 to F (hex.) are CP-specific.

Data Handling Blocks in the User Program 3

This chapter contains important information for working with data handling blocks in the user program and for using data blocks in different functions. It also offers information on determining the area length remaining for data transfer, the runtime of the data handling blocks, possibilities of reducing runtime and a host of additional information of importance to your programming staff.

3.1 Calling Data Handling Blocks

Calling DHBs in the Program

At which points in the program can you call handling blocks? The following is the simplest course of action:

1. Calling the SYNCHRON block only at initial start and in the cycle.
2. Calling the remaining blocks SEND/SEND-ALL/RECEIVE/RECEIVE-ALL/CONTROL/FETCH/ RESET only during cyclic program execution.
3. You can program events that you wish to be deleted during time-controlled or event-driven program processing and invoke a data handling block in such a way that only flags are set to begin with.

   During cyclic program execution, the respective data handling blocks will then be called.

Conditions for Calling DHBs

In principle, handling blocks can be called at any point within the user program, for example in the warm restart organization block or within time-controlled execution (OB 13, timed interrupt). However, the following limitations must be observed.

- The SYNCHRON DHB can only be called conditionally.
- The CPU 948R has the following three or four modes of program execution (among others)
  - Cyclic program execution (OB 1, lowest priority)
  - Time-controlled program execution (OB 6, OB 9, OB 10 to OB 18, priority is selectable)
  - Interrupt-driven program execution (OB 2 to OB 5 or OB 2 to OB 9, priority is selectable) and
  - "Soft STOP" with cyclic execution of communication jobs (OB 39).
- The data handling blocks cannot be interrupted. This must be taken into account when calling handling blocks with long runtimes (see Section 3.5).

In "soft STOP" (OB 39), handling blocks can be called irrespective of the point where the interrupt has occurred in RUN mode. For these purposes only, the required interfaces must have been initialized with a SYNCHRON at CPU restart for the "soft STOP" (OB 38) status.

3.2 Calling SEND-ALL and RECEIVE-ALL

For the SEND and RECEIVE blocks, the operating mode SEND-ALL or RECEIVE-ALL can be selected using job number "0" (see block description in Chapter 4). These blocks permit data transmission to be initiated and controlled by the interface.

The SEND-ALL and RECEIVE-ALL functions must be called "regularly" so that a request for communication from the interface is detected in time. For example, the following options and combinations are available:

- Call once (n times) per cycle and per interface (short response times)

  Call with a call distributor: in the first cycle for interface 1, in the second for interface 2, etc. (low cycle load)

- Call within time-controlled execution (OB 13 every 100 ms, independent of cycle time fluctuations)

- Call whenever other blocks do not have to be called owing to the process (evenly distributed cycle load and/or assignment of priority to control activities over communication)

- Call one ALL function after another until the first, second, nth etc. run is not an idling run. In the next cycle, continue with the following ALL function (evenly distributed cycle load).

3.3 Using Data Blocks

The data handling blocks work with up to five DB or DX data blocks in total:

1. A data block which has been assigned the SSNR, A-NR, ANZW and BLGR parameters indirectly. Here, the corresponding data block must be selected before calling the handling block
   (see Section 2.2 "Direct and Indirect Parameter Assignment").
2. a) A data block which has been assigned its source parameters or destination parameters indirectly
      (see Section 2.2 "Direct and Indirect Parameter Assignment").
   b) A data block which, with READ/WRITE, contains the source parameters and destination parameters.
3. A data word of a data block can be used as a PLC condition codeword.
4. The data to be transmitted/received may have to be taken from/stored in a DB.

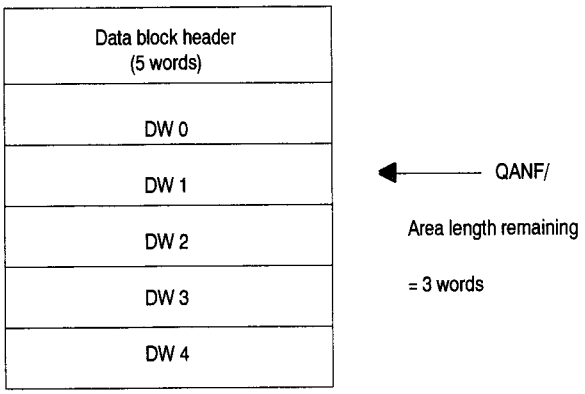
Here, care must be taken to ensure that all data blocks used exist and that they are long enough; the use of data blocks comprising more than 256 words is possible.

The use of DB 0, DB 1 and DX 0, DX 1 and DX 2 is permitted. If they are used, the handling block rejects them with an error indication.

3.4 Memory Organization, Area Limits

Area Length Remaining

The "area length remaining" is ascertained when executing the source/destination parameters in the handling block. Here, this means the difference between the length of an area (DB length, number of flag bytes, for example) and the preset QANF/ZANF start address.



The maximum quantity of data to be transmitted must not be greater than the area length remaining (legality check) or must be oriented to this area length remaining (area limit): joker length (joker length: see QLAE/ZLAE parameter); transmission of the length remaining is used by several CPs.

Naturally, the QANF/ZANF start address must exist. In the example (DB 17), the values from 0 to 4 would be permissible.

With all types of area, the rules for procedure set out above apply. The information blocks below contain additional information.

QTYP/ZTYP Counter Locations ZB, Timer Locations TB

The CPU 948R has 256 counter locations and 256 timer locations:

256 counter locations : C 0 to C 255
256 timer locations : T 0 to T 255

QTYP System Data Area BS

The CPU 948R has 256 system data values in the BS area:

256 system data values : BS 0 to BS 255

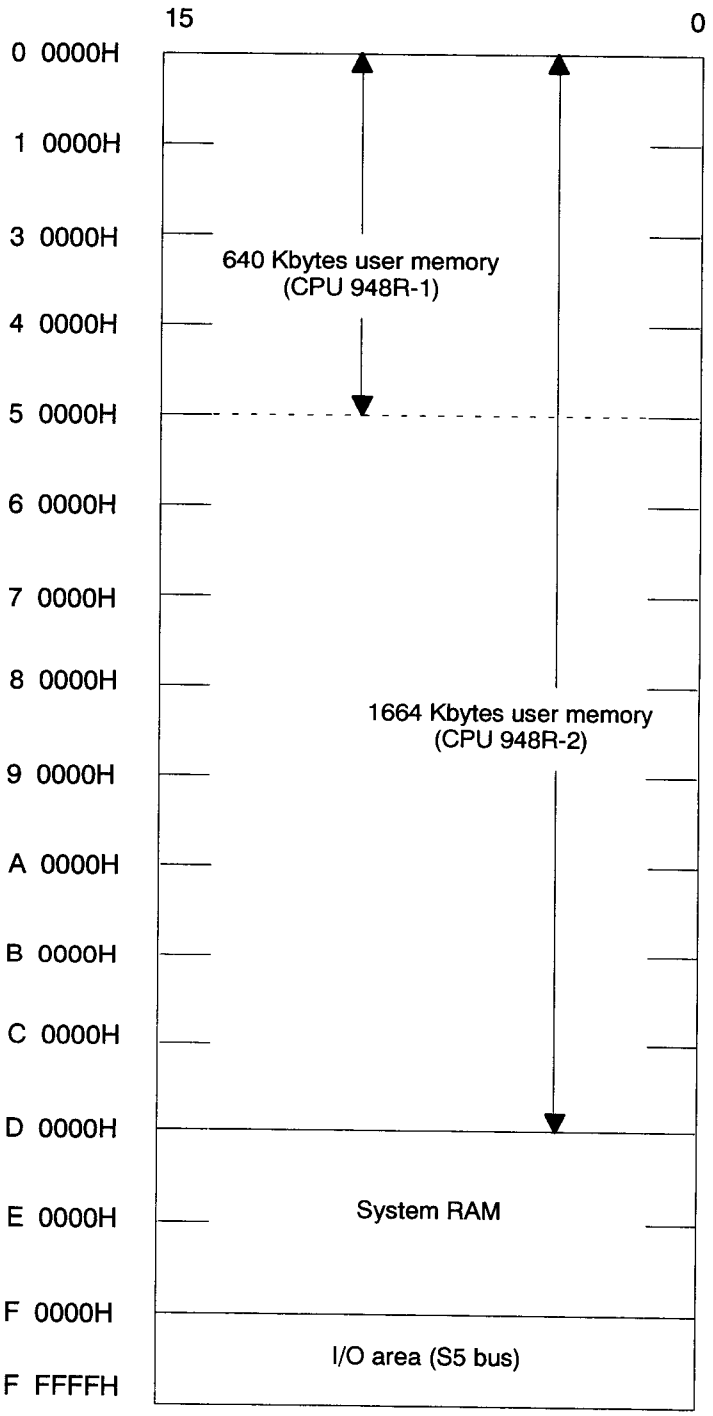
With the exception of BS 60 to BS 63, BS data may only be read by the user. BS data should therefore only be assigned the as QTYP parameter. Please see the Programming Guide for the CPU 948R for more detailed information on the BS (= system data area RS) data.

3

QTYP/ZTYP Absolute Addresses AS

With the CPU 948R, all the addresses that have been released for the user program area in the Programming Guide for this CPU are permissible AS addresses.

Overview of the address area of the CPU 948R:



The position of blocks in the RAM can be changed by means of the "Compress memory" function.

Special care must be taken when using the "absolute addresses" type. When accessing data blocks, for example, please note that these blocks will disappear from memory or change their position within memory when

- being generated (G DB and GX DX operations)
- the memory is compressed (by programmer input or automatically) or
- input/updates are carried out on the programmer.

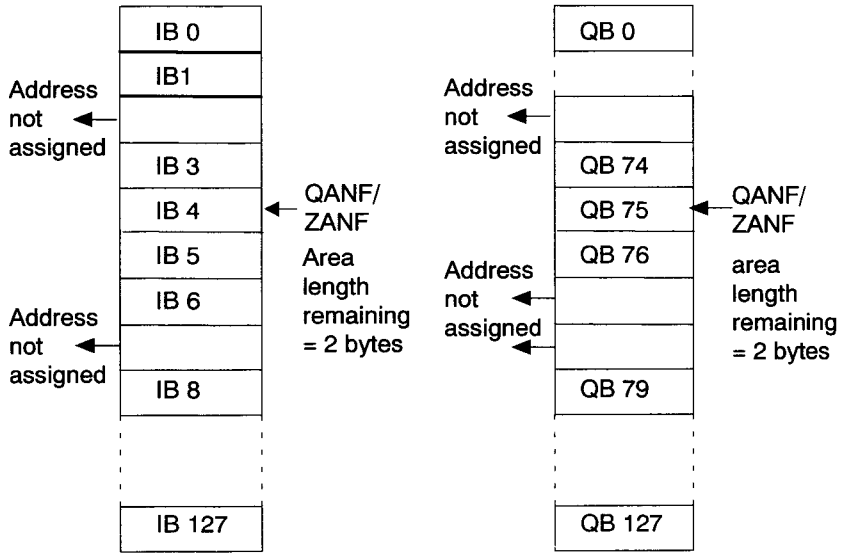
These changes are taken into account "automatically" by the data handling blocks when using the "DB" or "DX" types. Any errors do **not** result in illegal areas being accessed, but an error bit is set in the PAFE byte.

QTYP/ZTYP Flags MB

The CPU 948R has 256 flag bytes: FY 0 to FY 255

QTYP/ZTYP PIQ/PII (AB/EB)

The I/O area (PY) can be an incomplete area (with gaps) since normally not all input and output addresses are assigned or enabled. Example:



The area length remaining is the number of assigned addresses up to the next gap.

Note: RI and RJ areas cannot be specified. If these areas are to be transferred by means of data handling blocks, they must be copied into a DB.

3

3.5 Runtime

Runtime without Handshake

Runtime without Handshake	SEND	RECEIVE	FETCH	RESET	CONTROL
	idling				
CP in one-sided I/O	0.6 ms	0.7 ms	0.5 ms	0.5 ms	0.5 ms
CP in switched I/O	0.2 ms	0.2 ms	0.2 ms	0.2 ms	0.2 ms

Runtime with Handshake

As shown in the table below, the runtime of the data handling blocks comprises up to four components:

1. Basic runtime:

   The handling block requires this time, for instance, to prepare data transmission, update the ANZW, send parameters to, or receive parameters from, the interface, etc.

   These waiting times are interface-dependent (see CP description). However, the handling block aborts the function if the interface does not respond within a maximum waiting time (PAFE, "Interface not responding in time").

2. Waiting time A
3. Waiting time B

   } Waiting times of the handshake

4. Data transmission time:

   This time increases proportionally with the quantity of (net) data to be transferred (see SYNCHRON, parameter BLGR).

Runtime with Handshake	SEND	RECEIVE	FETCH	RESET	SYNCHRON
1. Basic runtime	2.3 ms	3.0 ms	1.6 ms	1.0 ms	5.1 ms
+	+	+	+	+	
2. Waiting time A					
+	0 ... 5 ms	0 ... 5 ms	0 ... 5 ms	0 ... 5 ms	0 ... 10 s
3. Waiting time B					
+	+	+	+	+	
4. Max. quantity of data to be transferred	0 ... 5 ms	0 ... 5 ms	0 ... 5 ms	0 ... 5 ms	0 ... 10 s
	512 (bytes)	512 (bytes)			
x	x	x			
Time per byte in the area:					
one-sided					
– MB	15 µs	35 µs			
– EB/AB	30 µs	65 µs			
– DB/DX/ZB/ TB/AS	2 µs	8 µs			
switched					
– MB	20 µs	40 µs			
– EB/AB	40 µs	70 µs			
– DB/DX/ZB/ TB/AS	4 µs	9 µs			

The table above illustrates that the data handling block user can influence the runtime of SEND and RECEIVE blocks by means of the number of bytes to be transmitted. The quantity of data required for a job (QLAE/ZLAE parameter) can be reduced when the send/receive data is

- checked critically
- subjected to reasonable limitations
- structured without gaps.

Whereas the QLAE/ZLAE parameters determine the quantity of data required for a **job**, the frame size specifies the **maximum** quantity of data per block call to be transmitted by the handshake (see SYNCHRON, BLGR parameter).

The "efficiency" of a SEND/RECEIVE block; that is, the relation

$$\frac{\text{Data transmission time (4.)}}{\text{Basic runtime (1.) + Waiting time A (2.) + Waiting time B (3.)}}$$

increases with the frame size.

Notes on the Runtime

The times listed apply for a one-sided CP plugged into the CC; the times apply for a switched CP installed 2.5 m from the CC.

"Runtime" is the processing time of data handling blocks.

You can save runtime by observing the following rules:

- Direct assignment of the SSNR, A-NR, ANZW and BLGR parameters is faster than assigning them indirectly.
- Direct assignment of the source/destination parameters is faster than assigning them indirectly (type XX).
- The use of data words in DB/DX is considerably faster than using input, output and I/O bytes.

Description: Data Handling Blocks

4

This chapter contains a representation of every available data handling block with block diagram, parameter table and detailed functional description.

4.1 SEND Function Block (FB 120)

Function Block

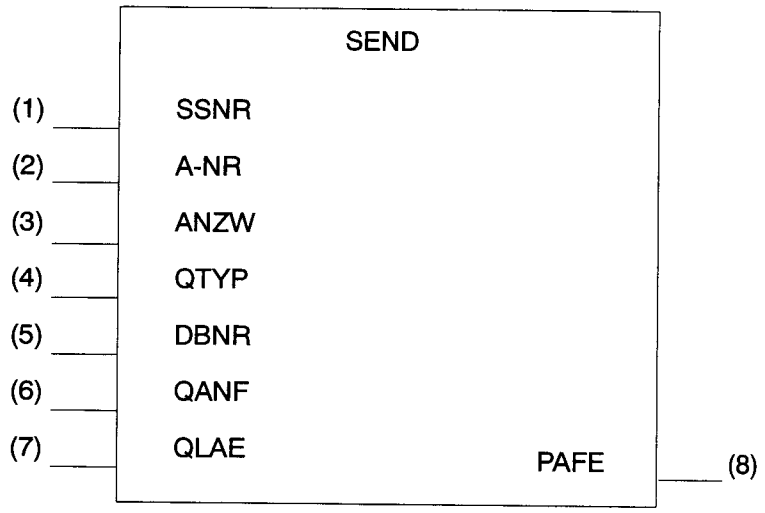


Figure 4-1 Block Diagram of the SEND FB

FB 120 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Number of the SEND/WRITE job
ANZW	I	W	Condition codeword, indicates the processing of the job
QTYP	D	KS	Type of data source (data block, flag, etc.) from which the data is transferred to the interface.
DBNR	D	KY	Number of the data block in the case of QTYP DB, DX, XX, RW; 64K area address with QTYP AS
QANF	D	KF	Relative start address of the data source
QLAE	D	KF	Quantity of source data (in bytes or words)
PAFE	Q	BY	Error bits

Function

The SEND block is used for sending of data and/or parameters from the CPU to the interface. The block has two operating modes:

- SEND-ALL
- SEND-DIRECT

The SEND-DIRECT function facilitates the "direct" initiation of a particular SEND (or WRITE) job determined by the job number (A-NR parameter). 1 to 223 are possible job numbers. This function can, for example, be used in particular process statuses (which the user program detects and evaluates) to output a message on printer.

The SEND-ALL function is selected with job number "0". This function checks whether the interface has a (SEND) communication request. If it has, the interface allocates the source parameters; that is, the interface determines which data is to be transmitted from the handling block, and for which job or for which job number this data is intended. Here, "all" job numbers can appear.

The SEND-ALL function is, for example, a simple means of having process images (process data) displayed on a monitor which can be updated regularly (CP 526). In this case, the CPU neither needs to know the image number (= job no.) which has just been selected, nor does it need to refresh the data, since the interface announces the communication request within the necessary time. The interface also fetches the data it needs for this display (and no others) from the next SEND-ALL called (see "Additional notes", "Calling SEND-ALL and RECEIVE-ALL").

SEND-DIRECT and SEND-ALL can/must also be combined if:

- the SEND-DIRECT is only being used to initiate a job – this is dependent on the parameters assigned to the data handling block and/or on the behavior of the interface during the handshake (see the CP description). The CPU transmits the data of this job to the interface via the SEND-ALL.
- the data length parameter (QLAE) assigned to the SEND-DIRECT is greater than the frame size. In this case, the interface independently calls for the subsequent blocks of this job from the SEND-ALL.

SEND-ALL

For the SEND-ALL function (job number = 0), the block requires the following parameters: "SSNR" interface number, "ANZW" (PLC) condition codeword, "PAFE" error byte.

The block receives the source parameters from the interface.

It also receives the job number which is stored in the ANZW PLC and which indicates which job the ALL function was active for. A zero means "idling"; that is, the interface has no SEND communication request.

SEND-DIRECT

A handshake with the interface to initiate a SEND job is only carried out if:

- "RLO = 1" is transferred to the function block, and
- the interface has enabled the job ("job is running" bit in the job status = 0).

Otherwise (if no handshake takes place) it is a case of idling. When the block is idling, only the condition codeword is updated.

Typical Application

In the cyclic program, process statuses are evaluated, and influence the RLO. A SEND-DIRECT function is then called unconditionally (JU FB). Consequently, the result of the logic operation "switches" the handshake on or off. Whatever the situation, a current copy of the job status exists in the ANZW.

For the SEND-DIRECT function, the block first requires the SSNR, A-NR ($\neq 0$), ANZW and PAFE parameters. The block only requires the source parameters if a handshake is included. The SEND function is performed in different ways, depending on the parameters assigned to the block:

- If the interface can take the data, the SEND block transmits all the data to the interface.

  However, if the interface signals that it only requires the parameters of the job, only the source parameters will be transferred to the interface. If the quantity of data to be transferred is greater than the declared frame size, only the parameters will be transferred to the interface with the first data frame.

  The interface requests the data or the subsequent frames of this job from the processor via the SEND-ALL function.

- If the "NN" identifier is entered in the QTYP parameter, the block accepts the source parameters from the interface, and transmits all the data or the first subframe. If the interface does not supply a set of parameters, it is a case of "job initiation without data transfer".

- Assigning parameters with QTYP = "RW" is called "WRITE". SEND-DIRECT transmits the source and destination parameters as well as the address of the condition codeword to the interface. The interface fetches the source data via SEND-ALL.

  In the case of the CP 143, the destination parameters, together with the data, are transmitted to the communication partner, which stores them as destination data at the position specified by the destination parameters (see the CP description).

4.2 SEND-A Function Block (FB 126)

Function Block

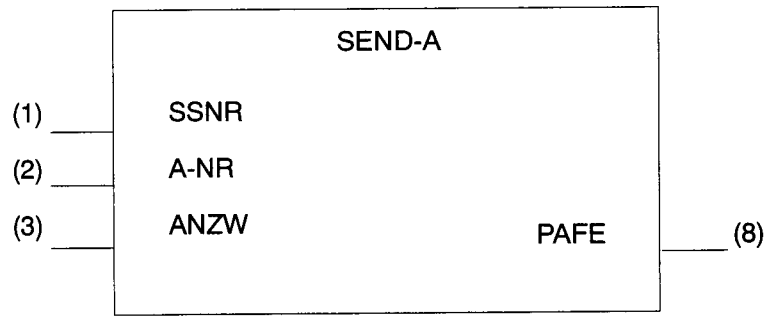


Figure 4-2 Block Diagram of the SEND-A FB

FB 126 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number: always 0
ANZW	I	W	Condition codeword: indicates processing of the job.
PAFE	Q	BY	Parameter assignment error: error indicators

Function

The SEND-A function block differs from the SEND function block by not having the QTYP, DBNR, QANF and QLAE parameters.

Whenever these parameters are irrelevant, the user of this block saves storage space as well as desk work and makes programs more transparent.

Further differences with reference to functionality or runtime do not exist.

4.3 RECEIVE Function Block (FB 121)

Function Block

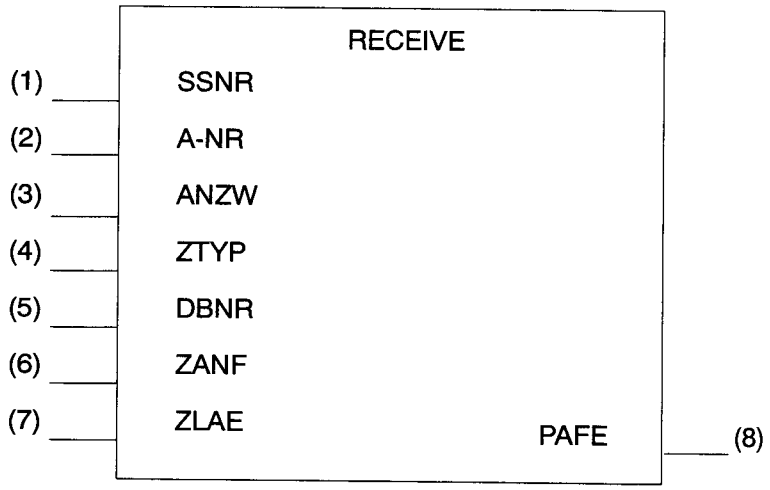


Figure 4-3 Block Diagram of the RECEIVE FB

FB 121 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number: number of the RECEIVE job
ANZW	I	W	Condition codeword: indicates processing of the job.
ZTYP	D	KS	Type of data source (data block, flag etc.), in which the data received from the interface is stored.
DBNR	D	KY	Number of the data block in the case of ZTYP DB, DX, XX, RW; 64K area address with ZTYP AS
ZANF	D	KF	Relative start address of the data destination
ZLAE	D	KF	Quantity of destination data (in bytes or words)
PAFE	Q	BY	Error byte

Function

The RECEIVE block is used for transferring data and/or parameters from the interface to the CPU. The block has two operating modes:

- RECEIVE-ALL
- RECEIVE-DIRECT

The RECEIVE-DIRECT function facilitates the "direct" initiation of a particular RECEIVE job determined by the job number (A-NR). 1 to 223 are possible job numbers. This function can, for example, be used to transmit data from an intelligent I/O module to the CPU (cf. IP 252 closed-loop control module).

The RECEIVE-ALL function is selected with job number "0". This function checks whether the interface has a (RECEIVE) communication request. If it has, the interface allocates the destination parameters; that is, the interface determines where data received from the handling block is to be stored and for which job or for which job number this data is intended. "All" job numbers can appear here.

The RECEIVE-ALL function is, for example, a simple means of transmitting input values (from a monitor with a keyboard) to the CPU. In this case, the CPU neither needs to know the image mask (= job no.) which has just been selected, nor whether input values are available or when they will be available, since the interface announces the communication request only when needed and the next RECEIVE-ALL called transmits the values to the CPU.

RECEIVE-DIRECT and RECEIVE-ALL can/must be combined if:

- the RECEIVE-DIRECT is only being used to initiate a job. This depends on the handling block parameter assignment and/or on the behavior of the interface during the handshake (cf. CP description). The data of this job will be transmitted from the interface to the CPU by means of the RECEIVE-ALL.
- the data length parameter (ZLAE) assigned to the RECEIVE-DIRECT is greater than the frame size. In this case, the interface independently supplies the subsequent frames of this job via the RECEIVE-ALL.

Owing to the runtime, only the data which is already in the CP can be transmitted by the RECEIVE function to the CPU.

Data which first must be requested by the CP via a bus link or which must be generated in some other way, is transmitted to the CPU by the combined action of the FETCH (see block description) and the RECEIVE-ALL function.

RECEIVE-ALL

For the RECEIVE-ALL function (job number = 0), the block requires the following parameters: "SSNR" interface number, "ANZW" (PLC) condition codeword, "PAFE" error byte.

The block receives the destination parameters from the interface. It also receives the job number, which is stored in the ANZW-PLC, and indicates which job the ALL function was active for. A zero means "idling"; that is, the interface had no RECEIVE communication request.

RECEIVE-DIRECT

A handshake with the interface to initiate a RECEIVE job is only carried out if:

- "RLO = 1" is transferred to the function block, and
- the interface has enabled the job ("RECEIVE job ready" bit in the job status = 1).

Otherwise (if no handshake takes place) it is a case of idling. When the block idling, only the condition codeword is updated.

If a RECEIVE-DIRECT function is called unconditionally (JU FB), the result of the logic operation "switches" the handshake on or off. In any case, a current copy of the job status exists in the condition codeword.

For the RECEIVE-DIRECT function, the block first requires the SSNR, A-NR ($\neq$ 0), ANZW and PAFE parameters. Only if a handshake is started does block require the destination parameters. The RECEIVE function performs in different ways, depending on the parameters assigned to the block:

- If the interface can provide the data, the RECEIVE block transmits all the data to the CPU. However, if (1) the interface signals that it only requires the parameters of the job, or if (2) the quantity of data to be transferred is greater than the declared frame size, only the destination parameters will be transferred to the interface. In the second case, the first data frame will also be accepted.

  The data or the subsequent frames of this job are transferred by the interface to the processor by means of the RECEIVE-ALL function. The parameters remain the same in any case for the user of the blocks. Only the time of data transfer is delayed in the cases just mentioned.

- If the "NN" identifier is written to the ZTYP parameter, the block receives the destination parameters from the interface, and transmits all the data or the first subframe. If the interface does not supply a set of parameters, it is a case of "job initiation without data transfer".

- The assignment of parameters with ZTYP = "RW", (READ/WRITE) is meaningless with the RECEIVE function and is not permissible.

4.4 REC-A Function Block (FB 127)

Function Block

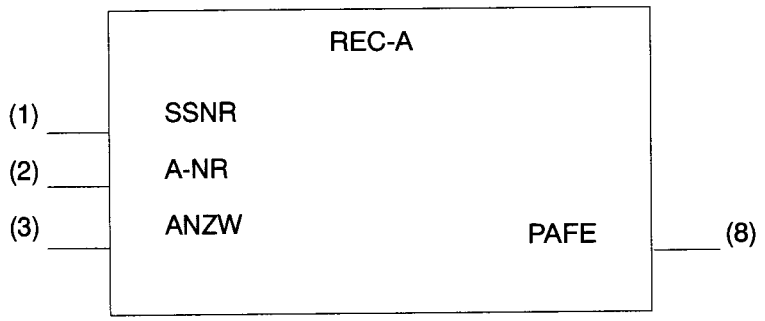


Figure 4-4 Block Diagram of the REC-A FB

FB 127 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number: always 0
ANZW	I	W	Condition codeword: indicates processing of job.
PAFE	Q	BY	Parameter assignment error: error byte

Function

The REC-A function block differs from the RECEIVE function block by not having the ZTYP, DBNR, ZANF and ZLAE parameters.

In cases where these parameters are irrelevant, the use of this block saves storage space as well as desk work and makes programs more transparent.

Further differences with reference to the functionality or runtime do not exist.

4.5 FETCH Function Block (FB 122)

Function Block

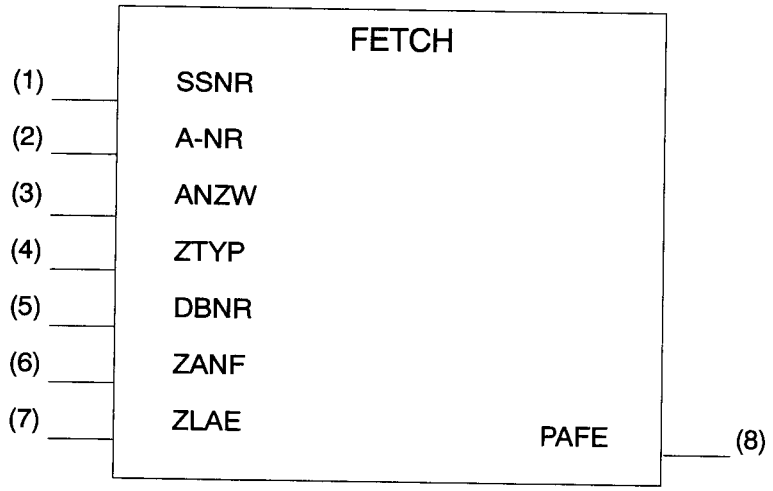


Figure 4-5 FETCH Function Block (FB 122)

FB 122 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number: always 0
ANZW	I	W	Condition codeword: indicates processing of job.
ZTYP	D	KS	Type of data destination (data block, flag, etc.) in which data transferred from the interface is stored.
DBNR	D	KY	Number of the data block in the case of ZTYP, DB, DX, XX, RW; 64K area address in the case of ZTYP AS
ZANF	D	KF	Relative start address of the data destination
ZLAE	D	KF	Quantity of destination data (in bytes or words)
PAFE	Q	BY	Parameter assignment error: error byte

Function

The FETCH block initiates a fetch job. This gives the CPU access to data which is not in the CP, but which must first be generated/acquired by the CP, for example, from another programmable controller by means of a bus link.

The FETCH block has only one operating mode:

- FETCH-DIRECT

1 to 223 are possible job numbers.

By transferring the job number, the destination parameters and the condition codeword address, the FETCH block informs the interface of the data required (job number), where this data is to be stored in the CPU (destination parameters) and which condition codeword this is to be indicated in. As soon as the data requested is in the CP, the interface provides the RECEIVE-ALL function with the parameters and the data. The FETCH block itself does not transmit/receive any data.

The handshake with the interface is only carried out if:

- "RLO = 1" has been transferred to the function block, and
- the interface has enabled the job ("job running" bit in the job status = 0).

Otherwise (if no handshake takes place) it is a case of idling. When the block is idling, only the condition codeword is updated.

If a FETCH-DIRECT function is called unconditionally (JU FB), the result of the logic operation "switches" the handshake on or off. In any case, a current copy of the job status exists in the condition codeword.

For the FETCH-DIRECT function, the block first requires the SSNR, A-NR ($\neq$ 0), ANZW and PAFE parameters. Only if a handshake is started does the block require the destination parameters. The FETCH function executes in different ways, depending on the parameters assigned to it:

- If the "NN" identifier is written into the ZTYP parameter, this is a case of "job initiation without parameter passing."
- The assignment of parameters with ZTYP = "RW" is referred to as "READ".

  In the case of the CP 143, this parameter assignment allows a READ job to be initiated (see the CP description).

The FETCH block transmits the source parameters and destination parameters as well as the address of the condition codeword to the interface. "Later", the interface transfers the data requested to the RECEIVE-ALL, which stores it at the position specified by the destination parameters.

4.6 CONTROL Function Block (FB 123)

Function Block

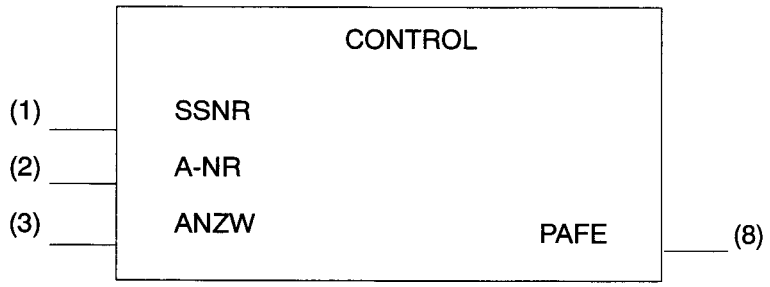


Figure 4-6 Block Diagram of the CONTROL FB

FB 123 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number of the job to be monitored
ANZW	I	W	Condition codeword: contains the result of the scan.
PAFE	Q	BY	Parameter assignment error: error byte

Function

The CONTROL block is used for scanning the status information of the interface. The block has two operating modes:

- CONTROL-ALL
- CONTROL-DIRECT

CONTROL-ALL

The CONTROL-ALL function (job number = 0) indicates in the low-order byte of the ANZW which job is currently being processed by the CP (or by the interface).

CONTROL-DIRECT

A job status exists in the interface for each job. It is managed by the interface and indicates, for example, whether a job is (still) running, or whether it has been completed without errors/with a particular error.

The CONTROL-DIRECT function transmits the job status selected with the A-NR parameter (job number 1 to 223) according to pattern 1 (see page II/2-24) to the condition codeword (see ANZW parameter).

4.7 RESET Function Block (FB 124)

Function Block

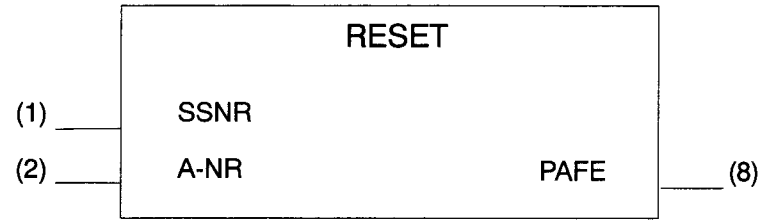


Figure 4-7 Block Diagram of the RESET FB

FB 124 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
A-NR	D	KY	Job number of the job to be reset
PAFE	Q	BY	Parameter assignment error: error byte

Function

The RESET block is RLO-dependent; the handshake with the interface is only carried out if "RLO = 1" has been transferred to the function block.

- RESET-ALL
- RESET-DIRECT

The RESET-ALL function (job number = 0) resets all jobs of this interface. It deletes all user data or interrupts all currently running jobs.

With the "direct" reset function (job $\neq$ 0), only the specified job of the interface is reset.

4.8 SYNCHRON Function Block (FB 125)

Function Block

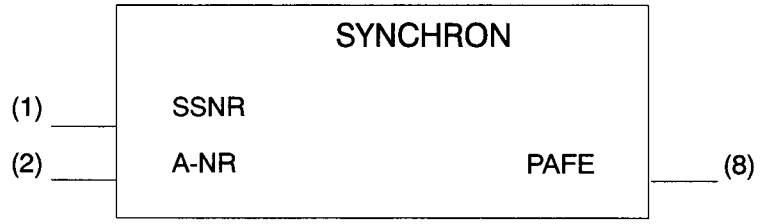


Figure 4-8 Block Diagram of the SYNCHRON FB

FB 125 Parameters

Parameter	Parameter Type	Data Type	Meaning
SSNR	D	KY	Interface number
BLGR	D	KY	Frame size
PAFE	Q	BY	Error byte

Function

The SYNCHRON block initializes the interface. The interface is deleted and preset, and the frame size is declared between the interface and the CPU. Every interface must be initialized on a "cold restart" (OB 20), in the "manual warm restart" (OB 21) and on a "cold restart with memory" (OB 22). If the "soft" STOP mode is also to be used, the SYNCHRON block must be called from the associated restart block (OB 38) for the required interfaces.

Examples of SYNCHRON DHB Calls on Restart

In the programming examples below, FB 125 is called depending on the evaluation of the H flag word.

CP switched:

```
STL                    Explanation
     :A     F0.2       Am master
     :JC    FB 125
NAME :SYNCHRON
```

SYNCHRON DHB (FB 125) call on restart (OB 20, 21, 22), CP one-sided, CP is in subunit A:

```
STL                    Explanation
     :A     F0.4       Am subunit A
     :JC    FB 125
NAME :SYNCHRON
```

SYNCHRON DHB (FB 125) call on restart (OB 20, 21, 22), CP one-sided, CP is in subunit B:

```
STL                    Explanation
    :A     F0.4        Am subunit B
    :JC    FB 125
NAME:SYNCHRON
```

Calling the SYNCHRON DHB (FB 125) Cyclically

If the interface has been passivated by an error (NAU..., for instance), you must depassivate and synchronize after correcting the error.

The identifier D0H is set in the PAFE byte for the duration of synchronization. FB 125 must therefore be called as long as this identifier exists.

```
STL                    Explanation
    :                  Program for CP 1
    :O F 10.0          F 10.0 must be set when
    :L KT 150.2        CP 1 has been repaired
    :SE T  1           Duration of depassivation max. 15 sec.
    :O(
    :L     FY 8        PAFE byte CP 1
    :L KH  0000
    :<>F
    :)
    :O F   10.0        F 10.0 must be set when
    :R F   10.0        CP 1 has been repaired
    := F   11.0
    :JC    FB 125
NAME:SYNCHRON
SSNR:      KY0.0       Page no. for CP 1
BLGR:      KY0.6
PAFE:      FY 8
    :A F   11.0        Depassivation requested?
    :A T   1           15 sec. not yet elapsed?
    :BEC               Yes ->BE because only 1 CP can be
    :                  inserted at any one time
    :A F   11.0        15 sec. not yet elapsed?
    :R F   11.0
    :JC    FBxx        Error message: CP 1 cannot be
    :L KB  0           depassivated
    :T FY  8
```

(Continued on next page)

```
STL                    Explanation
    :
    :                  Program for CP 2
    :O F   10.1        F 10.1 must be set when
    :L KT  150.2       CP 2 has been repaired
    :SE T  1           Duration of depassivation max. 15 sec.
    :O(
    :L     FY 9        PAFE byte CP 2
    :L KH  0000
    :<>F
    :)
    :O F   10.1        F 10.1 must be set when
    :R F   10.1        CP 2 has been repaired
    := F   11.1
    :JC    FB 125
NAME:SYNCHRON
SSNR:      KY0.2       Page no. for CP 2
BLGR:      KY0.6
PAFE:      FY 9
    :A T   1           Only one CP can be inserted at any one
    :BEC               time
    :
    :A F   11.1
    :R F   11.1
    :JC    FBxx        Error message: CP 2 cannot be depassi-
    :L KB  0           vated
    :T FY  9
    :
    :                  Program for CP 3
```

The FB is called once only via flag 10.0. The block call remains autonomous thanks to the "D0" identifier.

Only one interface may be synchronized at any one time. The interfaces are processed in sequence.

IMPORTANT:

Please note that only one SYNCHRON DHB call can be executed per cycle since, otherwise, errors would occur in the acknowledgement monitoring time between the CPU and the CP.

Frame Size

Declaration of the frame size takes place in such a way that the SYNCHRON block transfers a "desired" frame size to the interface corresponding to the BLGR parameter. This request is checked and, if necessary, **changed** by the interface (see the CP description). The "resulting" frame size forms the upper limit for the SEND and RECEIVE blocks and limits the maximum quantity of (net) data bytes to be transmitted/received per block call.

If the length of the area to be transmitted (QLAE/ZLAE) is larger, subsequent frames are transmitted by the SEND-ALL/RECEIVE-ALL functions. If, for example, a SEND-DIRECT is called with a QLAE = 70 bytes, and if the frame size is 32 bytes, the SEND-DIRECT transmits 32 bytes, the first SEND-ALL transmits 32 bytes and the second SEND-ALL transmits the remaining 6 bytes.

Table 4-1 Significance of the "Frame Size" Parameter

BLGR parameter	Frame with	
0	max. 256 bytes (default value)	Frame size desired
1	max. 16 bytes	
2	max. 32 bytes	
3	max. 64 bytes	
4	max. 128 bytes	
5	max. 256 bytes	
6	max. 512 bytes	
7 . . . 254	max. 256 bytes (default value)	
255	0 512 bytes (fixed)	

Select large frames in order to maintain high data transmission rates; low short runtimes of the data handling blocks require a small frame size (see runtime).

SIEMENS

SIMATIC S5

COM 155H Programmer Software for Configuring the S5-155H Programmable Controller

User's Guide (S5-155H, Part III)

C79000-B8576-C135-02

Preface (How to Use This Manual)

Working with COM 155H 1

Main Menu 2

Configuring and Initializing 3

Configuring the I/O 4

Error Diagnostics and Documentation 5

Copyright © Siemens AG 1994 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.

Technical data subject to change

Siemens AG
Automation Group
EWK Elektronikwerk Karlsruhe

6ES5 998-4SR21

Preface (How to Use This Manual)

This User's Guide describes the purpose and use of the functions of the COM 155H (CPU 948R) Configuring Package. This package not only enables user-friendly configuring of the S5-155H PLC, but also offers diagnostics and documentation functions. These are designed specifically with the requirements of the S5-155H in mind.

COM 155H comprises a screen form system for user prompting in the following situations:

- When configuring the S5-155H
- Fault diagnostics and error display in plaintext
- Documentation of the redundancy-specific part of the configuration.

The COM functions are grouped into screen forms according to task and can be activated in the relevant screen form on the programmer. The screen forms have a menu structure (see Chapter 1).

Target Group

This User's Guide is aimed at technicians, programming staff and maintenance personnel with general systems knowledge. If you have any questions which are not answered by this Guide, please consult your local Siemens representative.

Notes on the Contents

The information below concerning the contents of the individual chapters is designed to make it easier for you to use this User's Guide.

Chapter 1: Working with COM 155H

This chapter gives you information on the component parts, the scope of supply and all possible operator inputs (menu tree) of the COM 155H software package.

Chapter 2: Main Menu

Here, you are introduced to the main menu for your configuring work with all the relevant screen forms, including the input and output parameters.

In the main menu, you can define the data source (programmable controller or programmer) for the configuration. In addition, you have access to functions such as start/stop programmable controller, diagnostics, system handling with output of directories of contents and initiation of load and delete procedures.

Chapter 3: Configuring and Initializing

This chapter describes assignment of parameters to the operating system, the DB and DX data blocks and the I/O areas of the individual expansion units (EUs). There is also an explanation of how to define the address areas of one-sided and redundant I/O.

Chapter 4: Configuring the I/O

In this chapter, you will learn how to configure the I/O - digital input and output modules, analog input and output modules, CPs and IPs.

You will find out how to get from the main I/O basic screen to the individual I/O screen forms where you enter the configuring data for the digital and analog inputs/outputs and the CPs and IPs.

Chapter 5: Error Diagnostics and Documentation

This chapter gives you information on the possibilities and methods of searching for errors which can occur in the programmable controller after transfer of the configuration data. You can gain an overview of errors which have occurred and you can also obtain detailed information on each individual error.

The procedure for documenting your configuration is then described.

Index

The alphabetical index at the end of the manual will help you locate the most important terms in the manual.

Remarks Form

The (green) remarks form at the very end of the manual is provided for your comments and recommendations.

Reference Literature

See the Preface to "S5-155H Programmable Controller" (Part I) in this manual.

Contents

1 Working with COM 155H III/1-1

1.1 Installing and Operating COM 155H III/1-2
Configuration III/1-2
Installation III/1-2
Operator Prompting III/1-2
Exiting a Screen Form (V 3.x) III/1-3
Alternatives in COM 155H III/1-3

1.2 Installing and Starting COM 155H under "STEP 5 to Version 3.x" III/1-4

1.3 COM 155H Functions Under "STEP 5 to Version 3.x" III/1-5

1.4 Installing and Starting COM 155H Under "STEP 5 from Version 6.x" III/1-8
Starting COM 155H III/1-8

1.5 COM 155H Functions Under "STEP 5 from Version 6.x" III/1-9
Function Tree: COM 155H III/1-9
The Object Menu III/1-9
The Configuration Menu III/1-11
The Diagnostics Menu III/1-12
The Documentation Menu III/1-13

1.6 Explanation of Terms Used III/1-14

2 Main Menu III/2-1

2.1 The Main Menu III/2-2
F1: <CONF PLC> III/2-2
F2: <CONF FD> III/2-3
F3: <CONF PG> III/2-3
PLC Functions, F4 ..... III/2-4
Diagnostics, F5 ..... III/2-6
Defaults, F6 ..... III/2-9
System Handling, F7 ..... III/2-9
Configuring I/Os, F2 ..... III/2-16

3 Configuring and Initializing III/3-1

3.1 Configuring the S5-155H PLC (S5-DOS/Stage 6) III/3-2
Configuration Functions Screen Form III/3-2
Initializing the Operating System: <OS> III/3-3
Configuring the I/Os: <IOCONF> III/3-4

3.2 Initializing the Operating System
F1: <SYSTEM> (in the Initialize Operating System Basic Screen Form) . III/3-5
Operating System Parameters III/3-5
RAM/PIQ Comparison Error III/3-5
Operating System: Explanation of Parameters III/3-6

3.3 Initializing Activation of the Standby
F2: <TRAFDAT> (in the Initialize Operating System Basic Screen Form) III/3-9
Transfer Data for Activating Standby III/3-9

3.4 Initializing the I/O Areas
F3: <I/O 314> (in the Initialize Operating System Basic Screen Form) .. III/3-11
Area Parameters of the Switched I/O in the EU III/3-11

4 Configuring the I/O III/4-1

4.1 General Structure of the I/O Configuration Screen Forms III/4-2
I/O Basic Screen Form III/4-2
Displays in the Configuration Screen Form III/4-3
Processing the I/O Configuration Screen Form III/4-4

4.2 Structure of the Individual I/O Configuration Screen Forms III/4-6
Digital Inputs: F1 ..... III/4-6
Legend: DI III/4-8
Digital Outputs: F2 ..... III/4-9
Legend: DQ III/4-11
Analog Inputs: F3 ..... III/4-11
Legend: AI (Types 15 and 16) III/4-13
Analog Outputs: F4 ..... III/4-14
CP/IP: F5 ..... III/4-18

5 Error Diagnostics and Documentation III/5-1

5.1 Static Error Image of the I/Os III/5-2

5.2 Error Data Block III/5-5
Calling the Error DB III/5-5
Example of an Error Record Printout (Programmable Controller) III/5-6
Structure of the Error Diagnostics Screen Form III/5-6
Legend: Error DB III/5-7

5.3 Documenting with COM 155H III/5-8
Printing the Configuration Overview III/5-8
Printing the Configuration DB/DX III/5-9
Functions in the Print Menu III/5-9

Working with COM 155H

1

This chapter describes how to start the COM 155H software package, the two different versions of the main menu depending on whether you are using "S5-DOS/Level 5" of "S5-DOS/Level 6", and important terms used in the configuring and operator software.

1.1 Installing and Operating COM 155H

Scope of Supply and File Names

The scope of supply of COM 155H consists of

- a 3 1/2 in. diskette (for PG 635) and
- a 5 1/4 in. diskette (for PG 685/PG 7xx).

The diskettes contain the following files:

– S5PXC16X.CMD	COM 155H, Language selection command file
– S5PEC16X.DAT	COM 155H, Language selection data file
– S5OEC16X.CMD	COM 155H, Overlay English
– S5XEH01X.DAT	COM 155H, PLC H error message texts
– S5XEH02X.DAT	COM 155H, Help texts
– S5XEH03X.DAT	COM 155H Print headers,
– S5XEH04X.DAT	Programmer-PLC error messages,
– S5PEC16X.MEN	Menu description file Main Menu
– S5OEC16X.MEN	Menu description file Configuring
– S5PEC16X.OPT	Text file for including COM 155H in S5-DOS V6.x
– S5CR70ST.S5D	Standard function blocks: Handling - FB 120 to FB 127, analog input 2-channel - FB 40, analog input 3-channel - FB 43, redundant analog output - FB 41, error output via CP 523 - FB 48.
– AS511S01.DAT	AS 511 Protocol for S5-155H PLC.
– DB523EST.S5D	Error messages in English for CP 523

Configuration

The COM 155H software package runs on the following:

- The PG 635/685/7xx programmers
- AT-compatible PCs
  under S5-DOS, (STEP 5 Basic Package, Version 3.x or 6.x).

Installation

The installation depends on the STEP 5 version used (see Sections 1.2 and 1.4).

Operator Prompting

All COM 155H functions are activated via menu-driven operator prompting at the programmer.

COM 155H automatically detects the S5-DOS level under which it is started up.

- Under S5-DOS Version 3.x, COM 155H can be operated using the function keys and the keyboard, like all other programmer packages.
- Under S5-DOS Version 6.x, the COM 155H screens can be selected using drop-down menus instead of the function keys.

However, only the screen forms are selected using the drop-down menus; within the screen forms themselves, the function keys are used.

Selection of a softkey on the screen with the mouse has the same effect as pressing the corresponding function key on the keyboard.

If COM 155H has been installed under S5-DOS Stage 5, you can call COM 155H from the S5 command interpreter to access the following:

- The "Language Selection" screen form, if COM 155H has been installed in several languages. If not, skip this screen form.
- The "Defaults" screen form (see Figure 1-1).

  After setting the defaults (F3 <SELECT>), press F6 <EXEC> to reach the "Main Menu" from where you can branch to the individual submenus using the softkeys <F2> to <F7>.

If COM 155H has been installed under S5-DOS Level V 6.x, you reach the Main Menu by starting S5-DOS/ST (see Figure 1-4).

- You can use the "Change" function to exit STEP 5. Select COM 155H to reach the COM 155H Main Menu.

Exiting a Screen Form (V 3.x)

If you exit a screen form with <F8>, COM 155H responds in the following way:

- If there is no screen form in the higher-level hierarchy (that is, you have returned to a branch menu), the screen form contents remain on the screen but can no longer be changed.

  The branch menu is then displayed and processed. Press the key again to clear the screen.
- If the higher-level hierarchy also contains a screen form to be processed, this new higher-level screen form is displayed and processed.

Alternatives in COM 155H

In the configuring screen forms of COM 155H, there are fields containing alternatives for your selection. You can make your selection using F3 <SELECT>. Press F3 and then press it repeatedly to display each available alternative in turn. The alternatives field is not exited.

The COM 155H V3.0 can be used with all CPU 946R/947R and 948Rs. To convert the DX1 of a CPU 946R/947R to a DX1 of a CPU 948R:

- Select presets for 948R (PG may not be online to a CPU 946R)
- Load DX1 of the 946R/947R from FD.

  Reply to the query whether the DX1 of the 946R/947R should be set with NO. The DX1 is then converted to 948R format and is located in the PG memory.
- Save DX1 to FD.

1.2 Installing and Starting COM 155H under "STEP 5 to Version 3.x"

Installing on the Programmer with PCP/M

- Insert the diskette in drive A:.
- Copy the files on the diskette to user area 0 on the hard disk C:.

Installing on the Programmer with MS-DOS and STEP5 to V3.x

- Insert the diskette in drive A:.
- Copy the files to the STEP5 system directory on the hard disk C: (for example, PCOPY 0A: *.* C:\SIMATIC\S5_SYS).

Starting COM 155H

Make sure all files are available on the hard disk. The COM 155H package can only be called up from the S5 command interpreter (KOMI). The Defaults screen form is displayed:

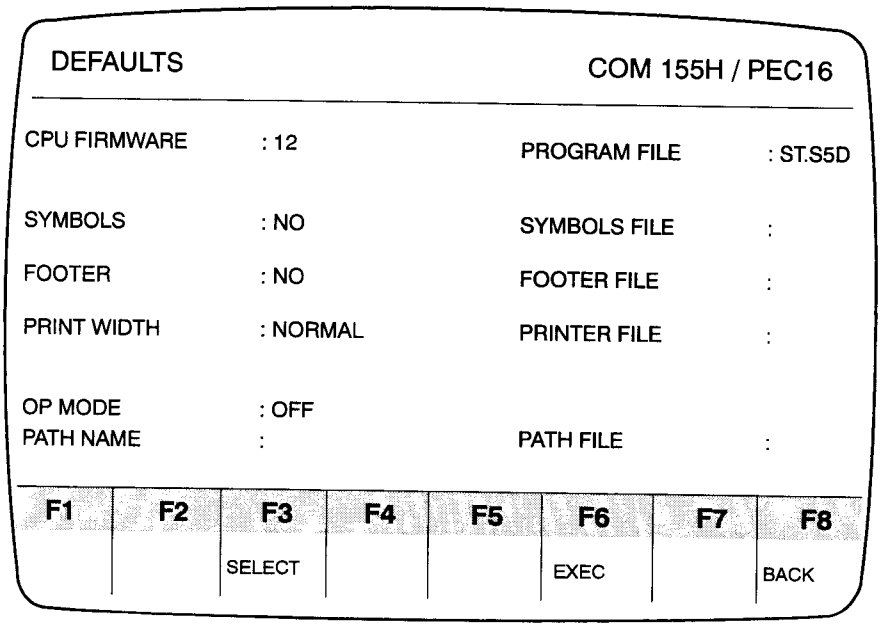


Figure 1-1 COM 155H Defaults Screen Form

- Press <SHIFT> + <Cursor to the right> or <Cursor to the left> to change from one half of the screen to the other. Press <Cursor up> and <Cursor down> to move the cursor up and down, respectively.
- If you position the cursor after the colon of the individual terms and then press F3 <SELECT>, you can enter or change the individual data easily.
- When you have entered all necessary data, press F6 <EXEC>. This enters the defaults and makes them immediately valid. The COM 155H Main Menu bar appears.
- If you want to exit the COM 155H package, press the <ESC> key.

1.3 COM 155H Functions Under "STEP 5 to Version 3.x"

Function Tree: COM 155H

The starting point for the COM 155H function tree is the "Main Menu" (see Figure 2-1), which you reach after entering the defaults.

The individual operator screen forms are arranged in the form of the tree shown in Figures 1-2 and 1-3. In each case, the relevant menu bar is shown with the key numbers and the functions which can be called.

- The softkeys shown horizontally at the top of the figures below describe the COM 155H Main Menu.
- There are follow-on screen forms for each of the functions selectable in the main menu. The functions offered by the follow-on screen forms can be seen from the menu bars shown.

  The call paths and dependencies are symbolized by connecting lines.
- If the configuration branch (F1/F2/F3 in the Main Menu) is exited, you must store the configured data on the destination medium (diskette or programmable controller) by confirming the acknowledgement message before returning to the Main Menu.
- Function key F4 <PLC FCT> in the Main Menu contains mechanisms for starting and stopping the S5-155H programmable controller. Special error indicators and diagnostics functions can be activated using the <DIAGN.> key (F5).
- The Main Menu function <SYSHAN> (system handling = F7) enables the display of the configured external I/O available in system DB1. Following this, selective or complete documentation of the configured data from the programmable controller or from the program file can be generated. In addition, you can initiate deletions in the programmable controller or on diskette (see Figure 1-3).

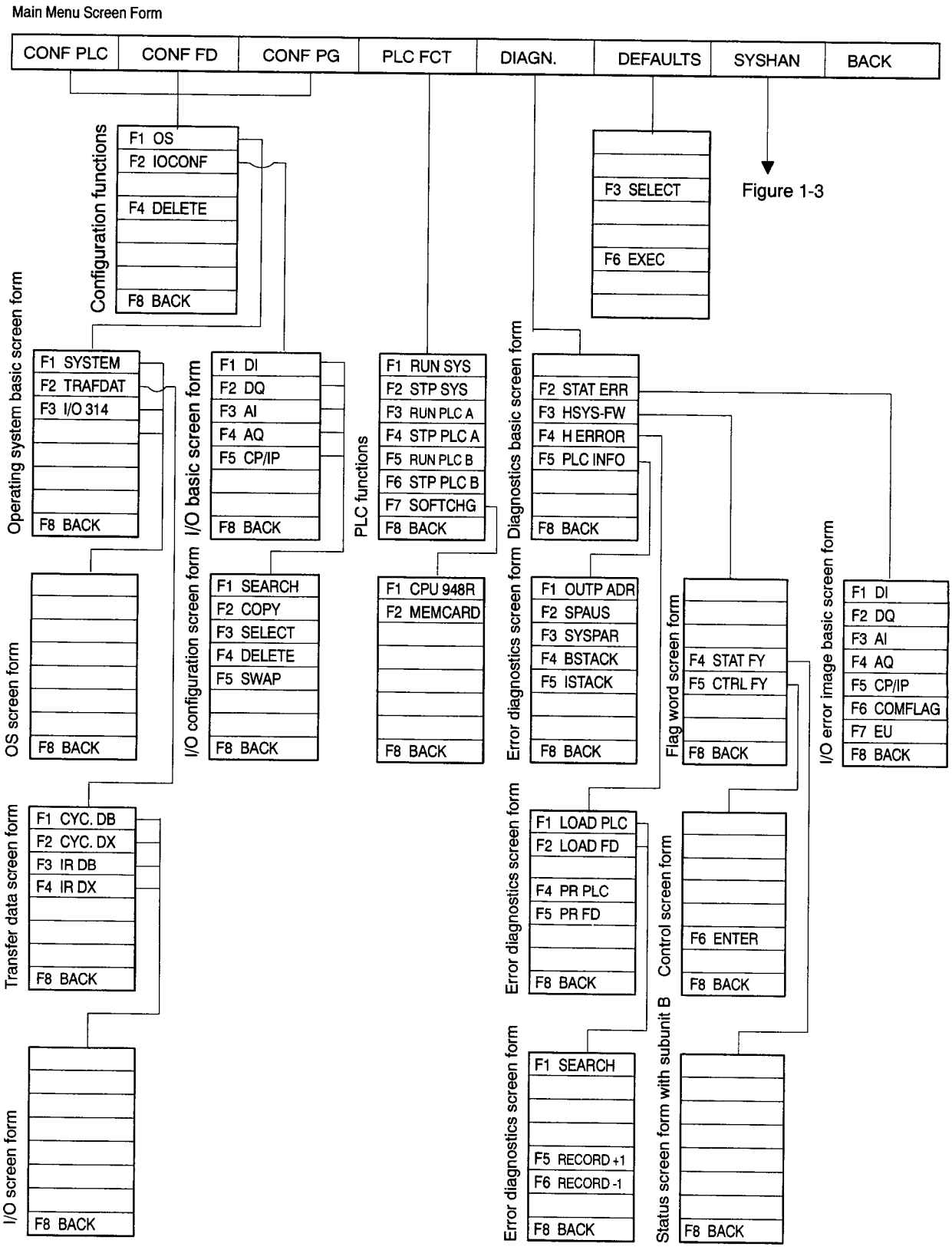


Figure 1-2 COM 155H Screen Form Tree for the S5-155H Programmable Controller (1)

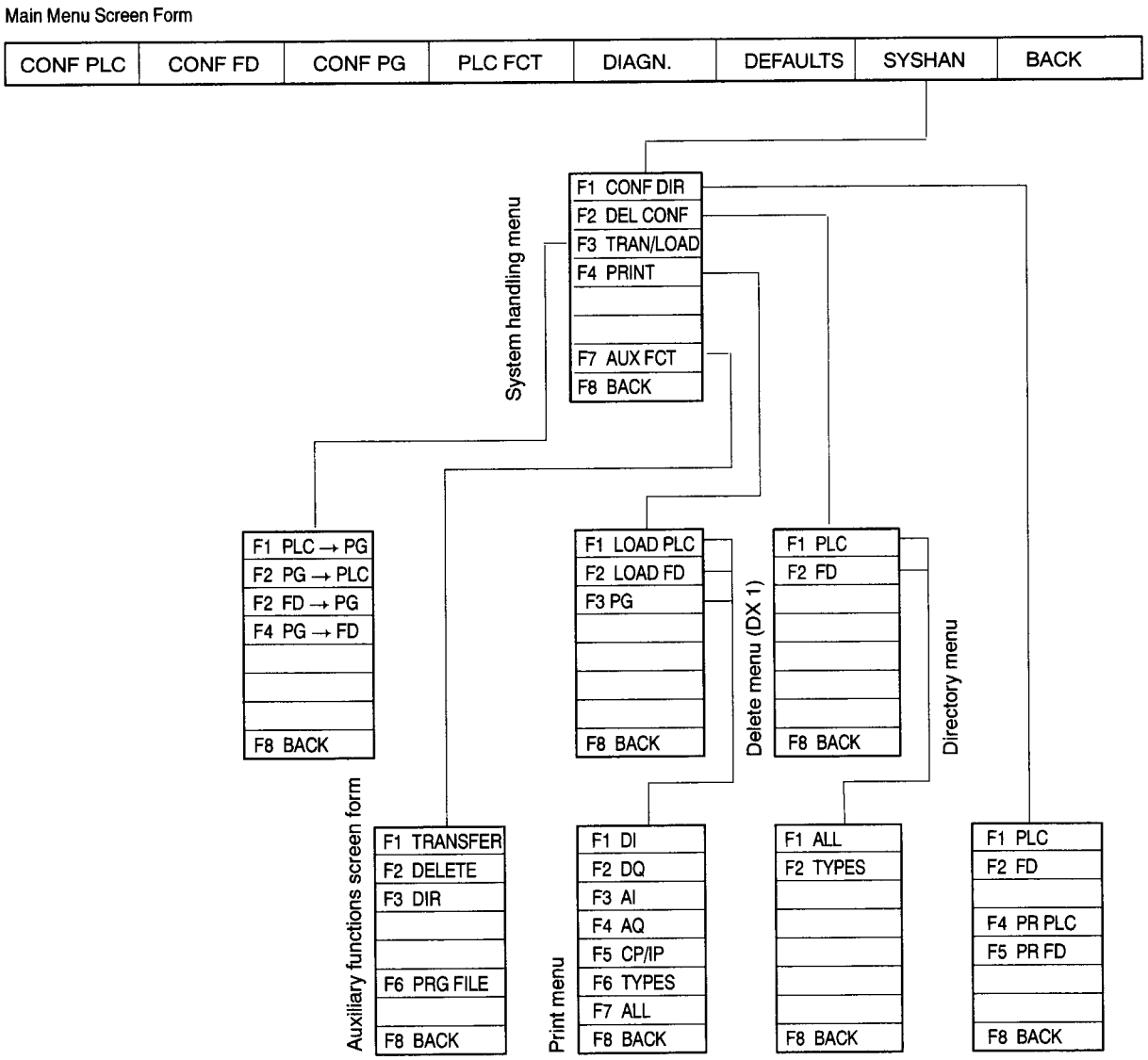


Figure 1-3 COM 155H Screen Form Tree for the S5-155H Programmable Controller (2)

1.4 Installing and Starting COM 155H Under "STEP 5 from Version 6.x"

Installing on Programmer with MS-DOS and STEP 5 from V6.x

1. Insert the diskette in drive A:.
2. Copy the files into the STEP 5 system directory on the hard disk (for example, PCOPY 0A: *.* C:\SIMATIC\S5_ST).
3. Change the logon file S5KES01X.OPT as described below:
   a) If there is not yet any software package loaded under STEP 5 V6.x, change the name of the S5PEC16X.OPT file supplied to S5KES01X.OPT, and copy this file also into the system directory.
   b) If there are software packages (GRAPH5, for example) already loaded, you must modify the S5KES01X.OPT file in your system directory:

      Load the S5KES01X.OPT file into the Editor and extend the user information by incrementing the "Number of subsequent element entries" by 1 and by entering the code for the COM 155H software package.

Relevant texts (ITEMS) are enclosed between inverted commas. Everything else is a comment. An element consists of three ITEMs, which must be within one line (max. 80 characters): Meaning of the element items:
-1:5 bytes BYTE [2–6] of the program name of the executable program
-2:1–3 program number analogous to the parameter assignment
in s5kes01x.men
-3:0 or 8 bytes of text to appear in the Change menu

File structure:	ITEM 1	Number of elements	
	ITEMS 2-4	ITEMS belonging to element	1
	ITEMS 5-7		2

Start of user information:

"1" Number of subsequent element entries for Level 6 applications

Program name	Program number	Text in CHANGE menu
"PXC16"	"101"	"COM 155H"

Starting COM 155H

Starting COM 155H from the STEP 5 menu bar:

- Beginning with the "Change" menu command, select the COM 155H software package entered there.

  The drop-down menus then contain a range of COM-specific supplementary functions, as well as the standard functions of STEP 5.

1.5 COM 155H Functions Under "STEP 5 from Version 6.x"

Function Tree: COM 155H

Figure 1-4 shows what the S5-DOS menu bar with the Main Menu and the individual drop-down menus look like after changing to COM 155H.

- The object-oriented or project-oriented defaults entered under S5-DOS Level 6 are accepted by COM 155H; that is, "Defaults" screen form cannot be called under Level 6.
- If you select an S5 standard function, the COM 155H user interface is suppressed and the relevant function is started. After completing the function, the COM 155H Main Menu bar appears again.

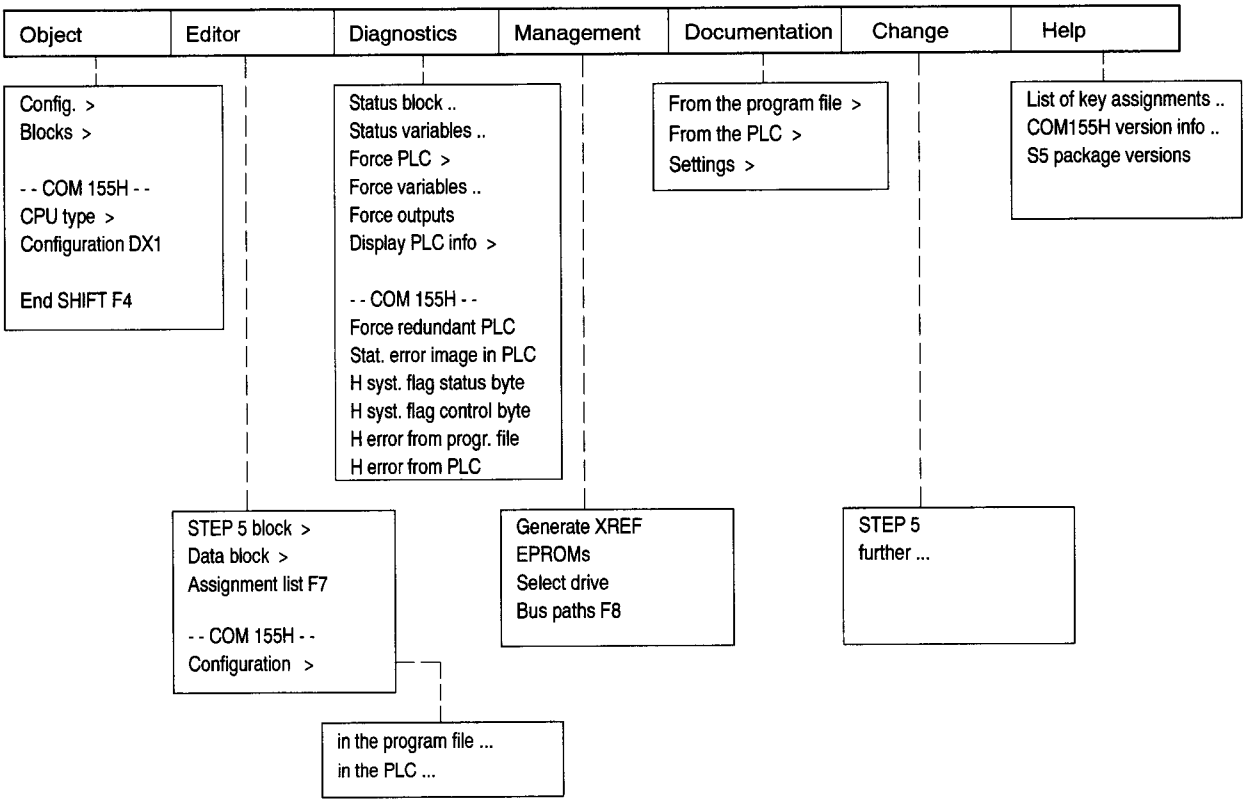


Figure 1-4 COM 155H Main Menu under S5-DOS/Level 6

The Object Menu

Figure 1-5 shows the functions available in the Object menu for displaying directories or handling data blocks and files, and how you can activate these.

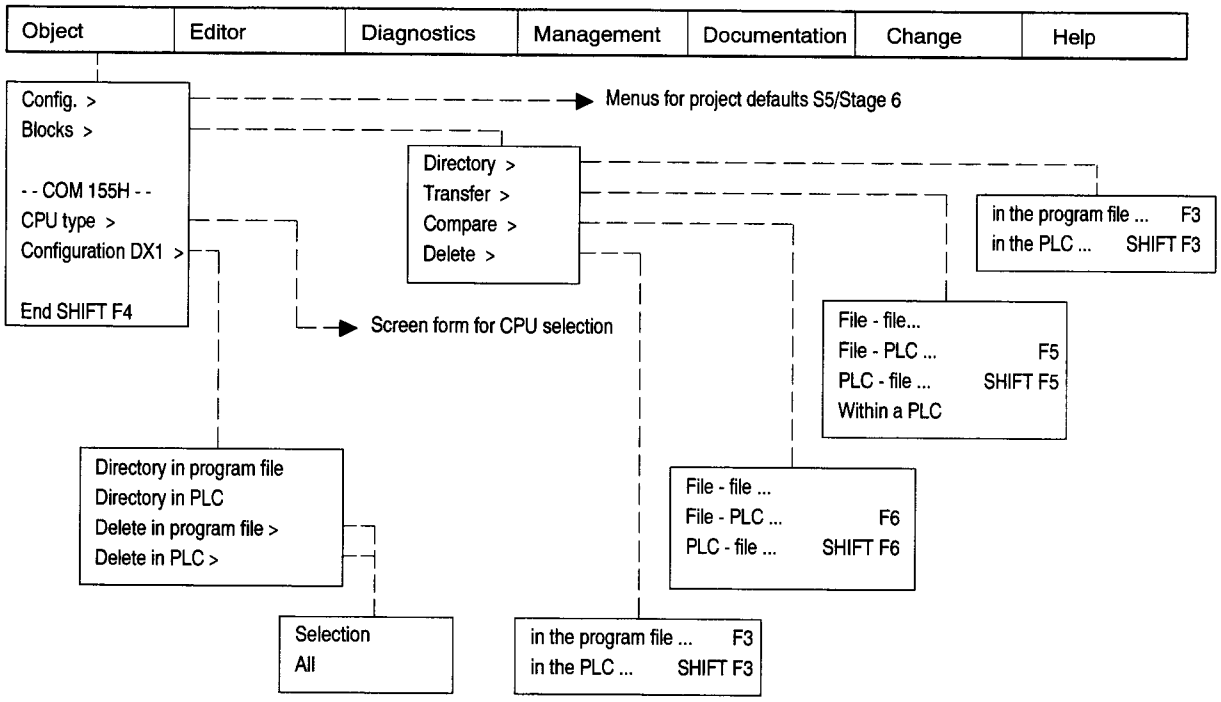


Figure 1-5 The COM 155H Object menu

CPU Type

You can choose between the following CPU types in the Selection screen form:

- CPU 946R up to firmware version 11
- CPU 948R up to firmware version 12
- CPU 948R

Configuration DX1

Selection of this menu entry calls the COM 155H directory functions.

"Directory in program file" or "Directory in PLC" gives you an overview of the configuration of DX1.

"Delete in program file" or "Delete in PLC" takes you via a Selection menu to the "Delete I/O" screen form, where you can select certain parts of the DX1 configuration for deletion.

The "All" function deletes the complete DX1.

The Configuration Menu

Figure 1-6 shows the COM 155H functions available in the Configuration menu for initializing the operating system and for configuring the I/O, and how you activate these functions.

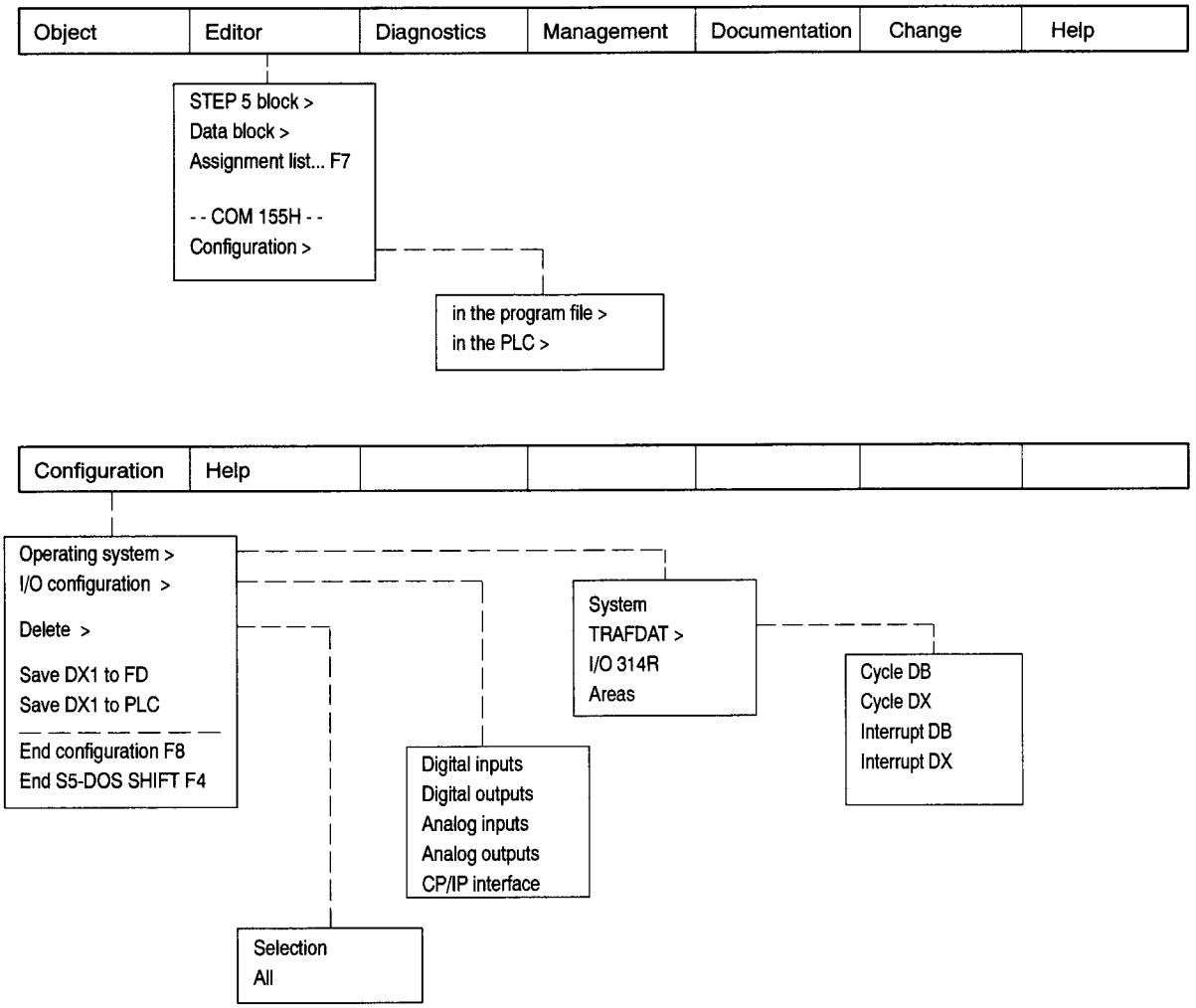


Figure 1-6 The COM 155H Configuration Menu

Initializing the Operating System

You reach the submenu for initializing the operating system via the "Program file/PLC" Selection menu.

- Select "System" to open the operating system basic screen form (see Figures 3-2 and 3-5).
- "TRAFDAT" takes you to the next submenu (see Figure 3-6) for selecting the screen forms for activating the standby (see Section 3.3).
- After selecting "I/O 314", COM 155H opens the screen form for entering the I/O areas for the switched I/O (see Figure 3-7).

Configuring the I/O

You reach the submenu for configuring the I/O via the "Program file/PLC" Selection menu.

- After selecting the relevant menu command, the associated I/O screen form is opened (see Figure 4-1).
- Pressing F3 <SELECT> in the softkey bar of the I/O screen form sets the relevant I/O type.

The Diagnostics Menu

Figure 1-7 shows which COM 155H functions you can activate for diagnostics purposes on the S5-155H in addition to the STEP 5 test functions.

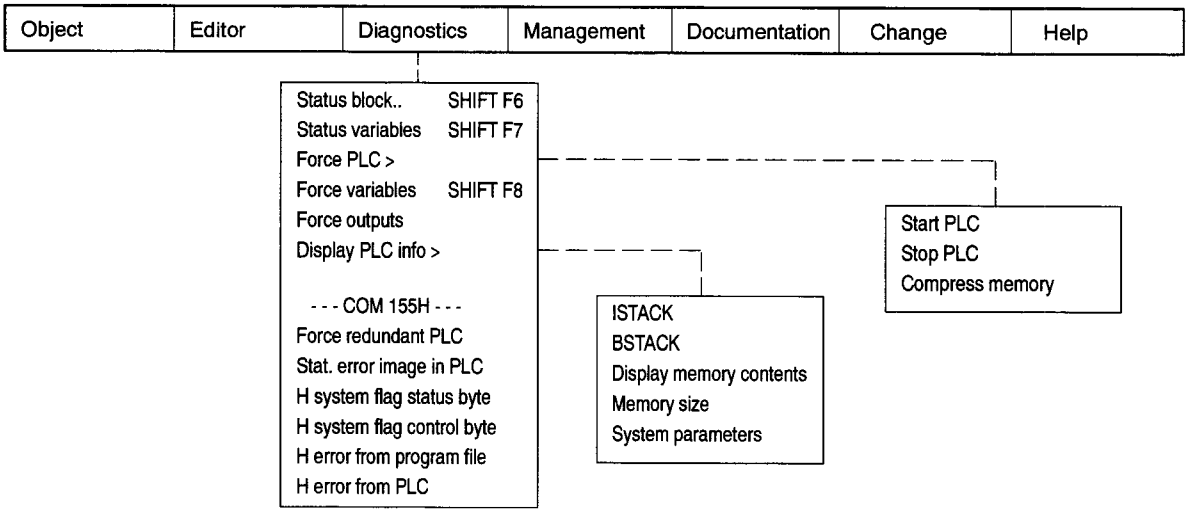


Figure 1-7 The COM 155H Diagnostics Menu

Activating Diagnostics Functions

After selecting one of the COM 155 functions in the Diagnostics menu, you reach the relevant processing screen form:

- **Redundant open-loop control**
  Takes you to the "PLC Functions" screen form (see Figure 2-2).
- **Static error image in PLC**
  Takes you to the "Static Error Image of the I/Os" screen form (see Figure 5-1).
- **H system flag status byte**
  Takes you to the "COM 155H Status Screen Form" (see Figure 2-6).
- **H system flag control byte**
  Takes you to the "COM 155H Control Screen Form" (see Figure 2-7).
- **H errors from program file**
  Takes you to the "Error Diagnostics" screen form (see Figure 5-5). The error DB configured in DX1 is loaded from the diskette or the hard disk and displayed on the screen.
- **H errors from the PLC**
  Takes you to the "Error Diagnostics" screen form (see Figure 5-4). The error DB configured in DX1 is loaded from the PLC and displayed.

The Documentation Menu

Figure 1-8 shows the print functions you can activate in COM 155H.

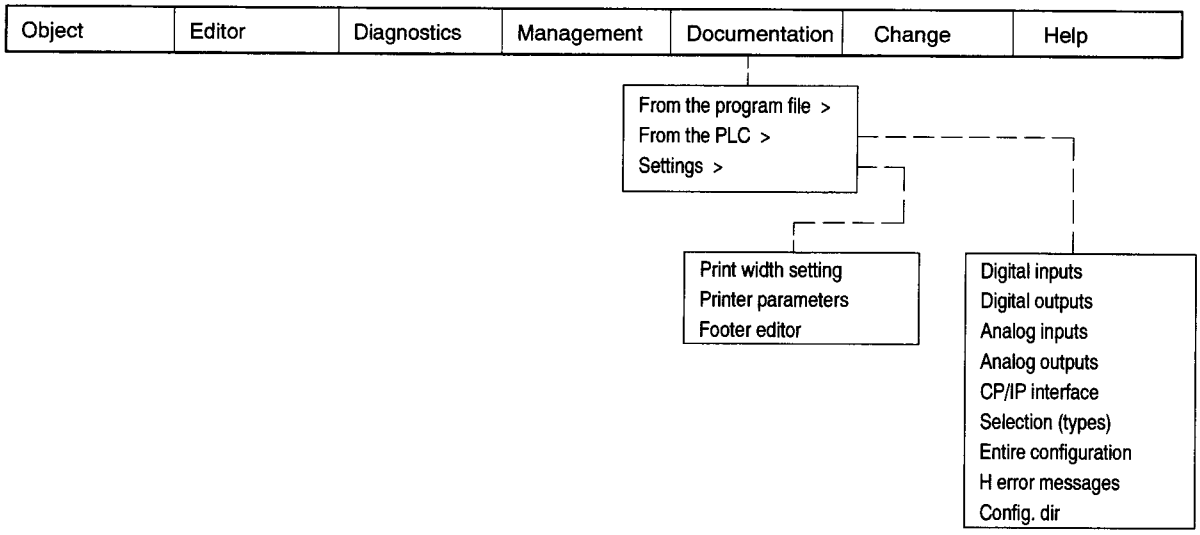


Figure 1-8 The COM 155H Documentation Menu

Activating Print Functions

You reach the submenu for print selection via the menu entries "From the program file" or "From the PLC", depending on the source desired.

- Depending on the menu command selected, all types of the selected I/O category are output to the printer in each case.
- After activating **Selection (types)**, you reach the "Print Menu" screen form in which you can select individual types for printing.
- If you activate the **H error messages** function, the reported errors located in the error DB of the selected file are printed out in compressed form.
- After selecting **Config. dir.**, an overview of the DX1 configuration is output.

1.6 Explanation of Terms Used

Firmware Version

In on-line mode, COM 155H determines the firmware version autonomously or signals if the wrong firmware version has been selected in on-line mode.

In off-line mode, the firmware version must be set to suit the version of the CPU 948R module.

Version xxx : Firmware version xx

Symbols

(see STEP 5 manual)

Footer

(see STEP 5 manual)

Print Width

You can specify three different print widths:

- Normal
- Narrow
- Extra narrow

Mode

(see STEP 5 manual)

Path Name

(see STEP 5 manual)

Program File

A program file must be set every time COM 155H accesses the hard disk or diskette (for example, load, transfer, delete).

Symbols File

(see STEP 5 manual)

Footer File

(see STEP 5 manual)

Printer File

(see STEP 5 manual)

Path File

(see STEP 5 manual)

Main Menu

2

This chapter explains the Main Menu with all associated screen forms including their input and output parameters.

2.1 The Main Menu

Main Menu Screen Form

The Main Menu appears after the defaults have been entered. The associated screen form looks like this:

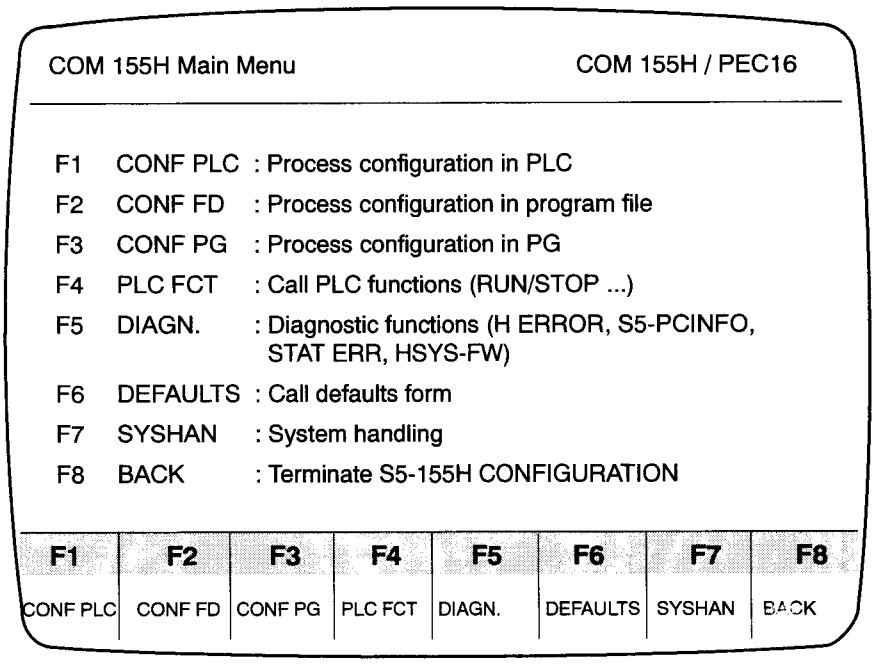


Figure 2-1 Main Menu Screen Form

You can exit COM 155H again at this point if you press F8 and answer the question "EXIT COM 155H?" by pressing the <INSERT> key.

Before you can execute the configuration or make any changes, you must specify the data source:

Key:	**F1**	**F2**	**F3**
Source:	**<CONF PLC>**	**<CONF FD>**	**<CONF PG>**

You can then execute the configuration or make changes (see Chapter 4).

F1: <CONF PLC>

PLC is the data source for the configuration

After pressing F1, you reach the Configuration screen form (only possible on-line). This loads DX1, if it exists, from the PLC; if it does not exist, the message "Data element does not exist" appears and an empty DX1 is generated in the programmer. This is then transferred to the PLC when you exit the configuration.

Changing the Configuration in RUN Mode

This function is required to permit defined selection of configuration changes even in the Solo and Redundant Modes of the S5-155H.

The following configuration changes are permitted:

- Cycle DB/DX
- Interrupt DB/DX
- Switched EU numbers
- Switched I/O in the address area FF080H...FF1FFH (analog and O area)

Sequence of the Functions

1. Changing the configuration in the switched I/O: First, plug in the new I/Os.
2. Make the desired configuration change with the usual introductory function "CONF PLC".
3. When the function has been completed, COM 155H detects that the PLC is in RUN mode and also detects the CPU type/firmware version.
4. COM 155H now checks that the differences between the DX in the PLC and in the programmer are permissible. If they are not permissible, the message "Change not possible - switch PLC to STOP" appears. If the changes are permissible, COM 155H transfers the DX1 to the programmable controller. All other changes can only be transferred to the programmable controller when it is in the STOP mode.
5. Start the depassivation function (control H flag word).
6. Update your user program only after you have done this.

F2: <CONF FD>

The FD is the data source for the configuration

After pressing F2, you reach the Configuration screen form. This loads DX1, if it exists, from the file; if it does not exist, the message "File does not exist" or "Data element does not exist" appears, and an empty DX1 is generated in the programmer. This is then stored on FD when you exit the configuration.

F3: <CONF PG>

The programmer is the data source for the configuration

After pressing F3, you reach the Configuration screen form. This loads DX1, if it exists, from the programmer memory; DX1 must then be stored with the help of the SYSHAN - TRAN PLC/FD function before you exit the configuration.

You need this selection if you have selected CONF PLC and if you have inadvertently ignored the message "Set PLC to STOP" before exiting the configuration.

PLC Functions, F4 <PLC FCT>

By pressing F4 in the main menu, you can have COM 155H execute the following functions regardless of which subunit your programmer is connected to:

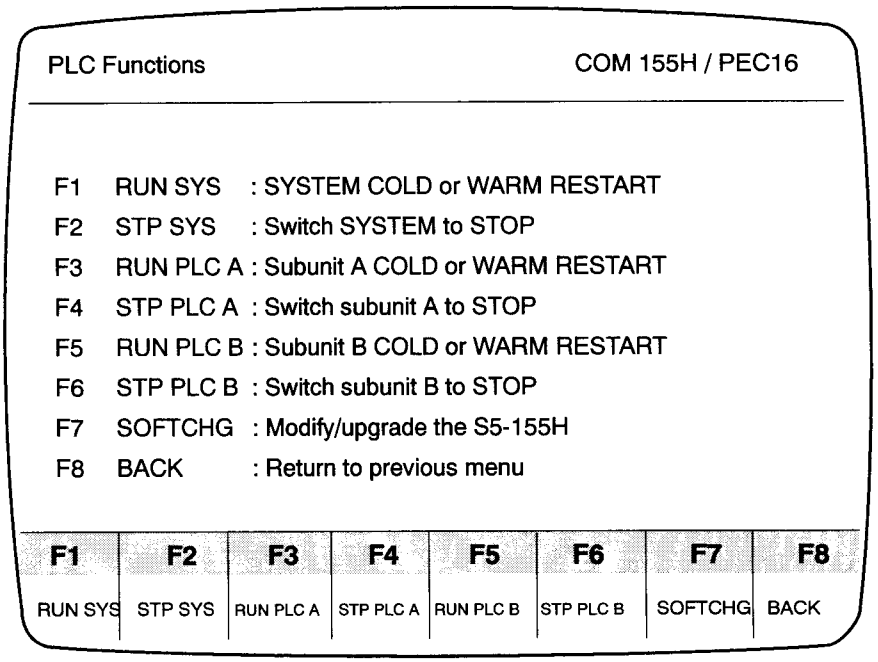


Figure 2-2 PLC Functions Menu

F1 <RUN SYS>:
The complete S5-155H PLC performs a restart routine (invocation of OB 20/OB 21).

After pressing F1, the following message appears:

"COLD (C)/WARM (W) RESTART/ABORT (BREAK):
WARM RESTART IS COLD RESTART WITH MEMORY RETENTION!"

Press the key for the desired restart mode or press <ESC>.

F2 <STP SYS>:
The complete S5-155H enters the STOP mode.

After pressing F2, the following message appears:

"SYSTEM STOP?"

Press <INSERT> for "YES" or <ESC> for "NO".

F3 <RUN PLC A>:
Subunit A performs a restart routine.

After pressing F3, the following message appears:

"COLD (C)/WARM (W) RESTART/ABORT (BREAK):
WARM RESTART IS COLD RESTART WITH MEMORY RETENTION!"

Press the key for the desired restart mode or press <ESC>.

F4 <STP PLC A>:
Subunit A enters the STOP mode.

After pressing F4, the following message appears:

"PLC A STOP?"

Press <INSERT> for "Yes" or <ESC> for "No".

F5 <RUN PLC B>:
Subunit B performs a restart routine.

After pressing F5, the following message appears:

"COLD (C)/WARM (W) RESTART/ABORT (BREAK):
WARM RESTART IS COLD RESTART WITH MEMORY RETENTION!"

Press the key for the desired restart mode (C or W) or press <ESC>.

F6 <STP PLC B>:
Subunit B enters the STOP mode.

After pressing F6, the following message appears:

"PLC B STOP?"

Press <INSERT> for "Yes" or <ESC> for "No".

F7 <SOFTCHG>:
You can modify/upgrade the CPU or the memory card without interrupting program execution.

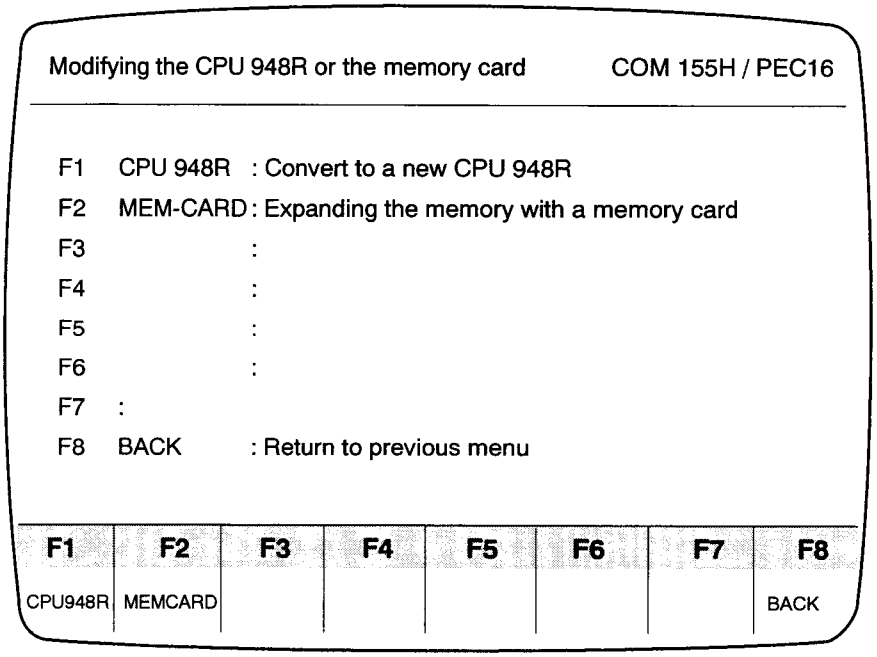


Figure 2-3 Modifying the CPU or the Memory Card

Diagnostics, F5 <DIAGN.>

Press F5 <DIAGN.> in the COM 155H Main Menu to reach the Diagnostics menu.

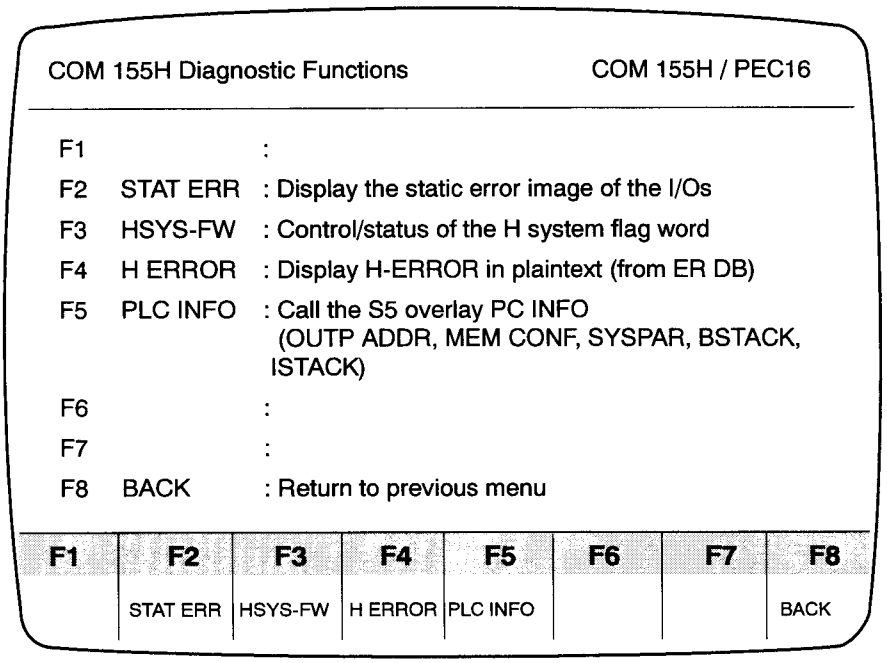


Figure 2-4 Basic Diagnostics Menu

F2 <STAT ERR>:
This key enables you to have a static error image of the I/O displayed and sorted according to the following criteria:

- Digital inputs
- Digital outputs
- Analog inputs
- Analog outputs
- CPs/IPs
- Interprocessor communications input flags
- Interprocessor communications output flags
- Expansion units.

Please refer to Chapter 5 "Error Diagnostics and Documentation".

F3 <HSYS-FW>:
Press this key to select either the status byte or the control byte of the H flag word. You can read out the current status information in plaintext and set the control information.

If you press F3 <HSYS-FW> in the Diagnostic Functions screen form, the following form appears on the screen:

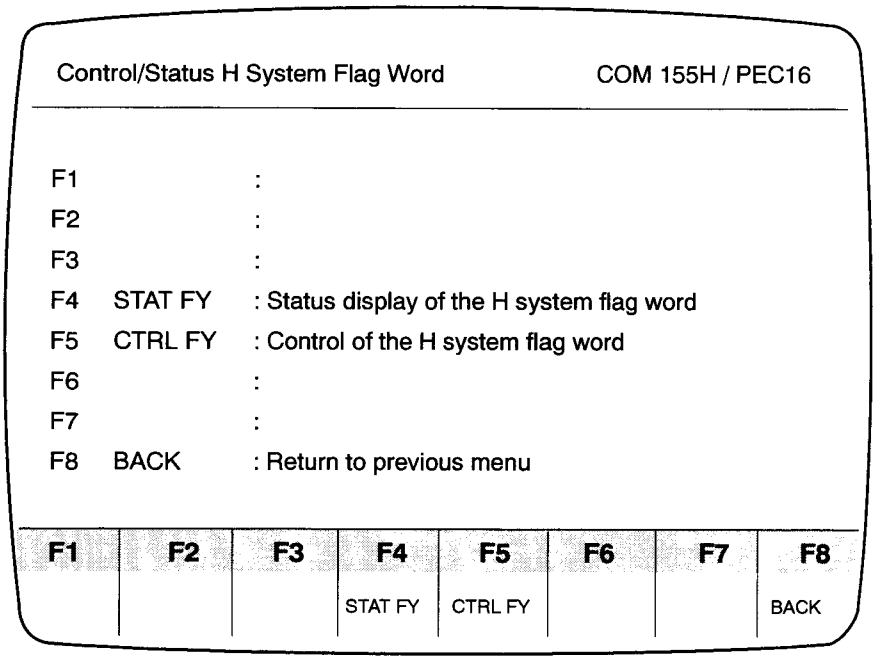


Figure 2-5 Flag Word Menu

Press F4 to display the status byte (low-order byte of the H flag word;) FY 0, for instance.

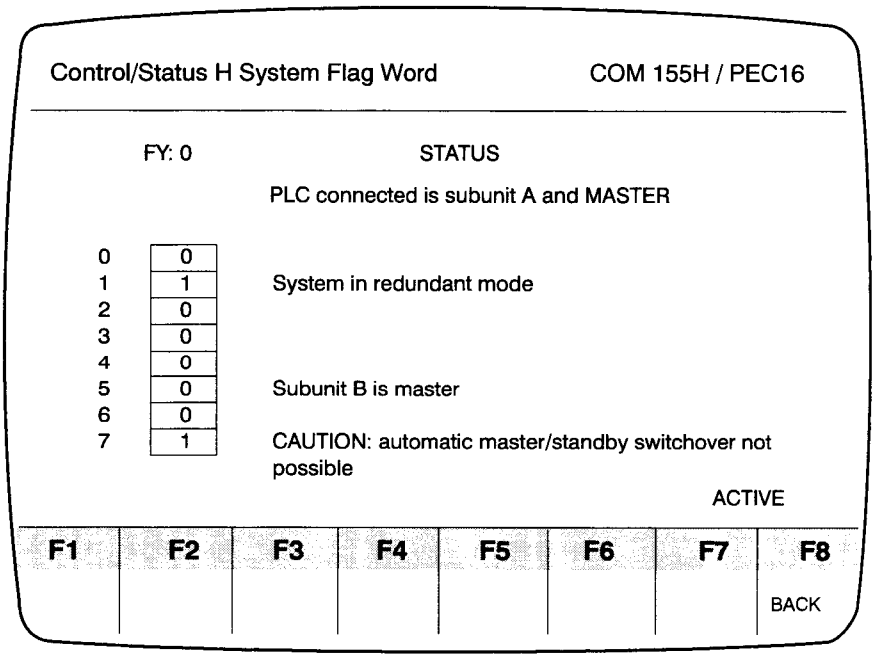


Figure 2-6 Status Form Showing Subunit B in STOP Mode and Programmer Connected to Subunit A

2

This form shows the following constellation:
Subunit B is in STOP mode and the programmer is connected to subunit A.

The bits concerned are set to "1" and their meanings are explained in plaintext. This form is dynamic; that is, all changes to the status byte are shown immediately.

Select F5 <CTRL FY> in the COM 155H Flag Word Menu to display the control byte (high-order byte of the H flag word;), FY1, for instance.

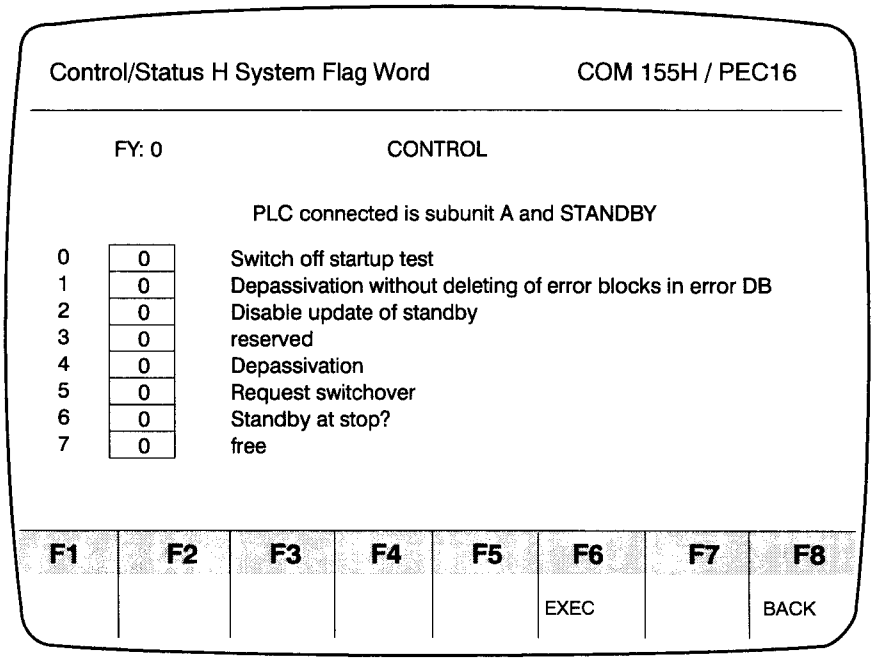


Figure 2-7 Control Form

Select the desired bit with the <Cursor up> and <Cursor down> keys. The associated texts appear in inverse video as long as the bit is "0". As soon as you overwrite the 0 with a 1, the text appears in normal representation. Make the change with F6 <EXEC>.

F4 <H ERROR>:
Please refer to Chapter 5 "Error Diagnostics and Documentation"

F5 <PLC INFO>:
Press this key to call the S5 overlay "PLC INFO" (see STEP 5 manual).

You can execute the following functions through the submenus:

- "Output address"
- "Memory size"
- "System parameters"
- "BSTACK"
- "ISTACK"

Please also refer to the S5-155H Instructions ("H flag word") for further information on the status and control bytes.

Defaults, F6

<DEFAULTS>

F6 <DEFAULTS> in the COM 155H Main Menu takes you to the Defaults screen form (see "Starting COM 155H" in Section 1.4).

System Handling, F7

<SYSHAN>

Press F7 <SYSHAN> in the COM 155H Main Menu to reach the "System Handling" menu.

The functions you can call from there refer to the configuration data block DX1! (Exception: "Transfer to the PLC" function and "AUX" function).

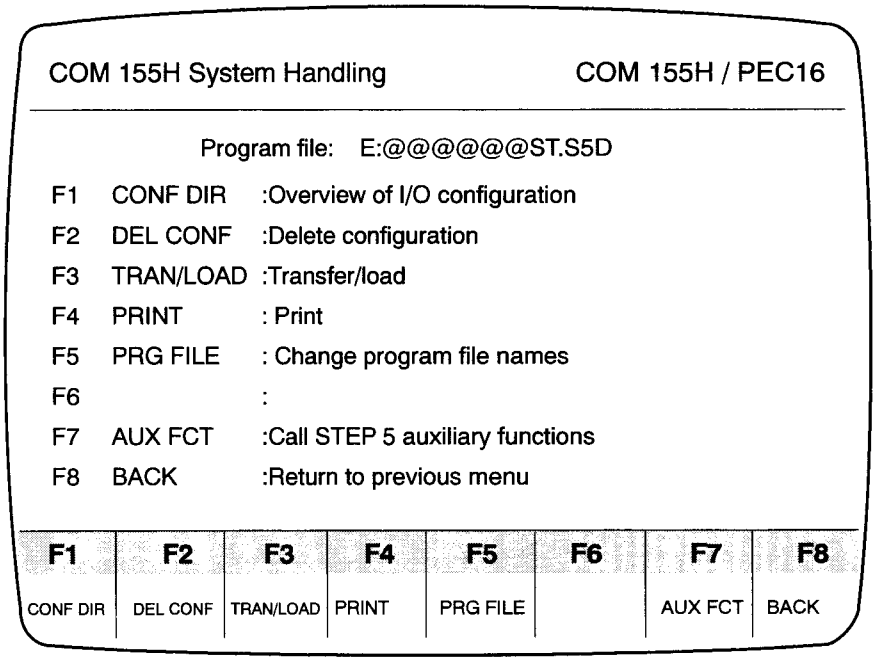


Figure 2-8 System Handling Menu

The individual functions are explained more closely below.

Make sure that a program file has been set before accessing (loading, transferring, deleting) the hard disk or the diskettes (see <F5> in the System Handling menu).

F1 <CONF DIR> (in the System Handling menu):

When this key is pressed, the Directory menu appears. This gives you an overview of your system configuration displayed on the screen. This overview can be read out from the **PLC** or from a **diskette (FD)** (see <F1> and <F2> in the directory).

If you require an overview of the configuration from the memory card, you must load DX 1 to diskette (via the S5 command interpreter EPROMs function) and then have the contents of DX 1 output from diskette (CONF DIR FD).

2

If access is not possible, an error message appears on the programmer screen.

If the function can be executed correctly, the overview of configured types is loaded and displayed on the programmer in the form of a list.

- <BACK> takes you back to the Directory menu.

You can also have this overview of your system configuration printed out from the programmable controller or diskette (<F4> or <F5>). Please refer to Section 5.3 "Documenting with COM 155H".

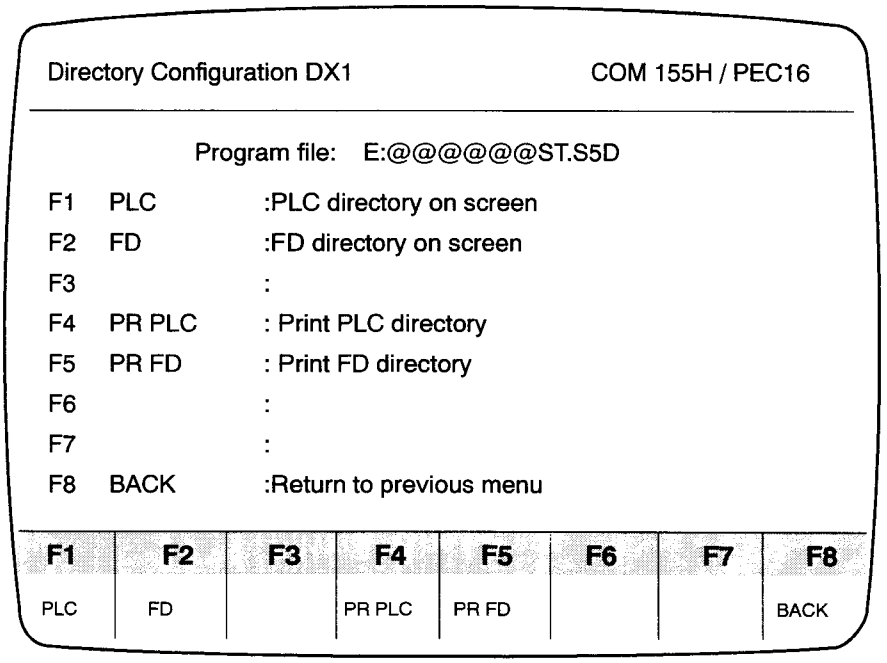


Figure 2-9 Directory Menu

F2 <DEL CONF> (in the System Handling Menu):

This key takes you to the I/O Delete menu. The Delete function refers to the DX 1 data block which has been loaded into the programmer. It deletes particular types (<F2>) of the configured digital or analog inputs and outputs, CPs/IPs or the entire system configuration (<F1>).

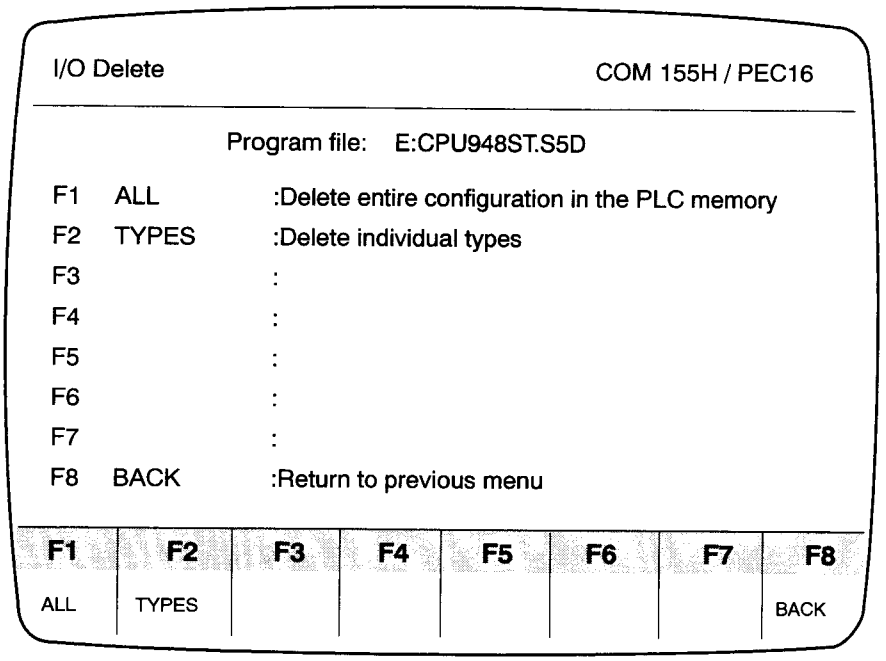


Figure 2-10 Delete Menu (DX 1)

When F2 in the Delete menu is pressed, a type matrix appears on the screen:

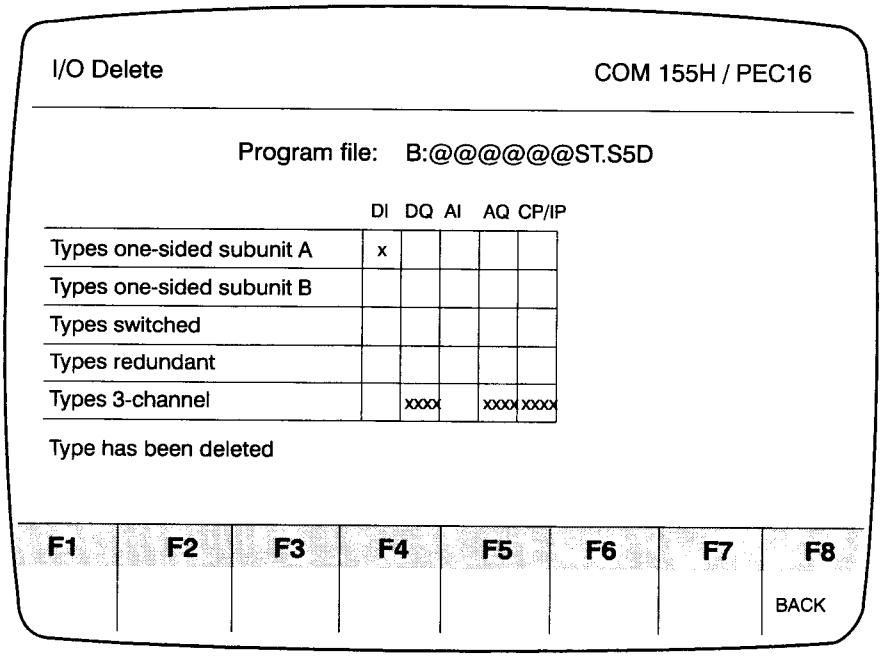


Figure 2-11 Delete Menu (Types)

- Move the cursor to the field of the type you wish to delete and press <INSERT>.

The message "Type has been deleted" appears on the screen to confirm what you have done, provided the type you have selected has also been configured.

Otherwise, the message "Type not configured" appears.

F3 <TRAN/LOAD> (in the System Handling menu):

When <F3> is pressed, the Transfer/Load menu appears on the screen.

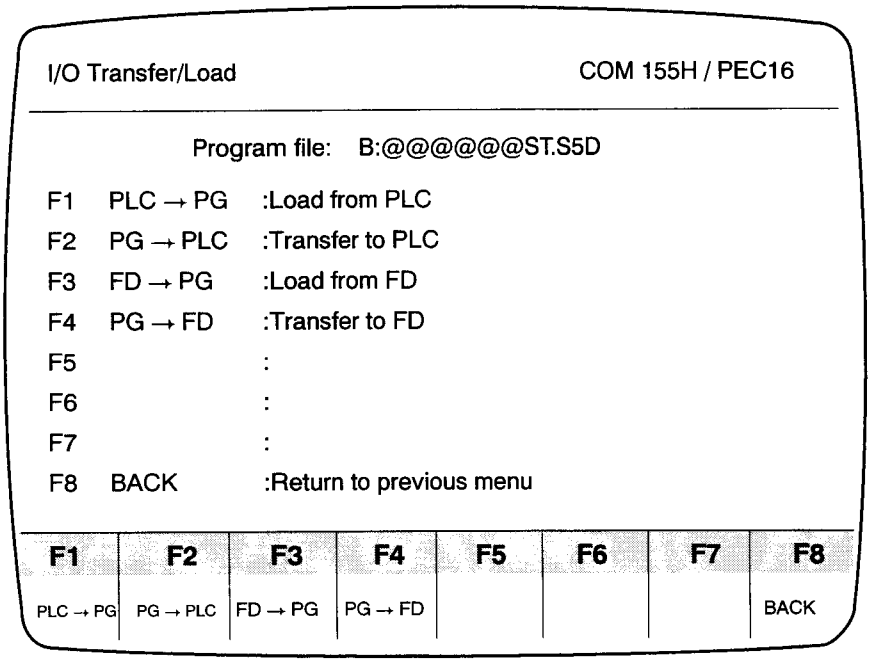


Figure 2-12 Transfer/Load Menu

<F1> <F3>: Loading in the Transfer/Load menu

The load functions enable you to read DX 1 from the programmable controller or diskette (FD) into the RAM of your programmer.

- When you press one of the load keys, the prompt "Load configuration (DX 1)?" appears.
- Confirm by pressing <INSERT> or press <ESC> for "No".

During the load operation, the "ACTIVE" message appears on the screen. On completion of the load operation, the message "Configuration (DX 1) has been loaded" appears.

If DX 1 does not exist, an appropriate error message is displayed.

F2, F4 <LOAD FD> <TRAN FD> (loading in the Transfer/Load menu):

The transfer functions enable you to transfer data from the programmer RAM to the programmable controller or diskette.

Press <F2> to reach the "Transfer to PLC" submenu. Here, you can decide whether you wish to transfer

- only DX 1 from the programmer memory using <F1>, or
- only the STEP 5 user program (all blocks except DX 1 and RAM DB) from the selected program file with <F3>

to the programmable controller.

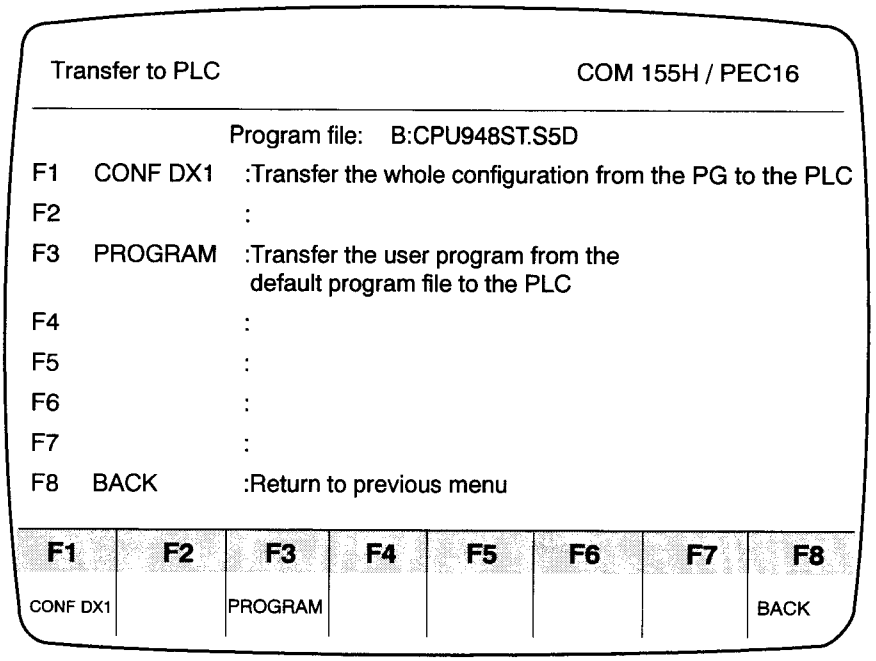


Figure 2-13 Transfer Menu

While DX 1 is being transferred, the "ACTIVE" message appears on the screen and while the user program is being transferred, the following message appears:

"BLOCK xx yyy BEING TRANSFERRED".

If DX 1 already exists, the following question appears:

"OVERWRITE CONFIGURATION (DX 1) IN PLC/FD?"

- Press <INSERT> for "YES" and the existing DX 1 will be overwritten. If you press <ESC>, the DB transfer will not be executed.

If DX 1 is to be transferred to the programmable controller but the connected subunit is not in STOP mode, the following message appears in the event of an impermissible "RUN change":

"DX 1 change not possible – switch PLC to STOP"

F4 <PRINT> (in the System Handling menu):

First, load or determine via the screen form (Figure 2-14) the configuration data block to be used.

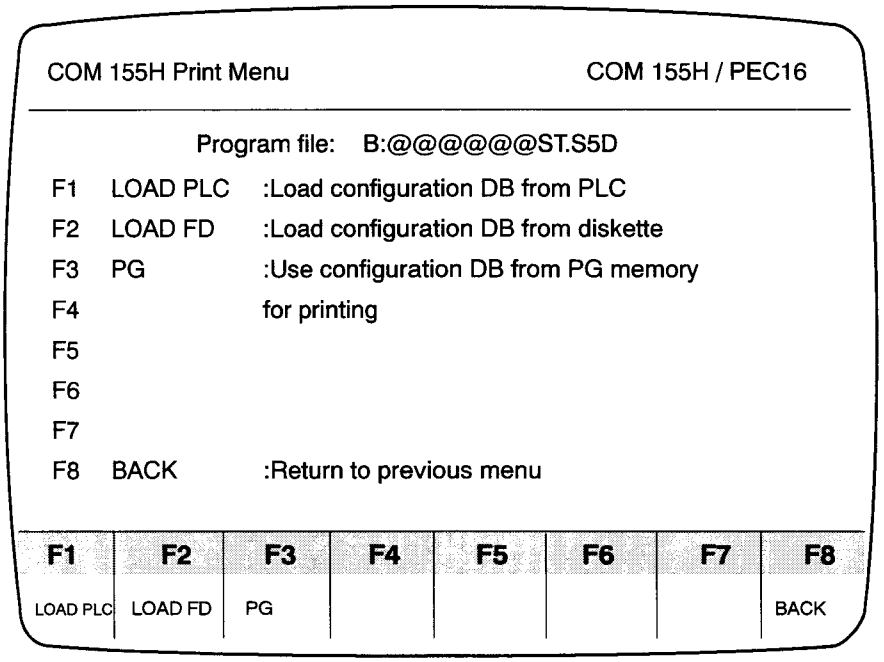


Figure 2-14 Print Menu

With the help of the Print functions (Figure 2-15), you can now

- Print all types of a particular category of configured I/O (DIs, DQs, AIs, AQs, CPs, IPs: using <F1> to <F5>)
- Print a particular type (for instance, "switched I/O") through all categories with <F6>
- Print the entire DX 1 with <F7>.

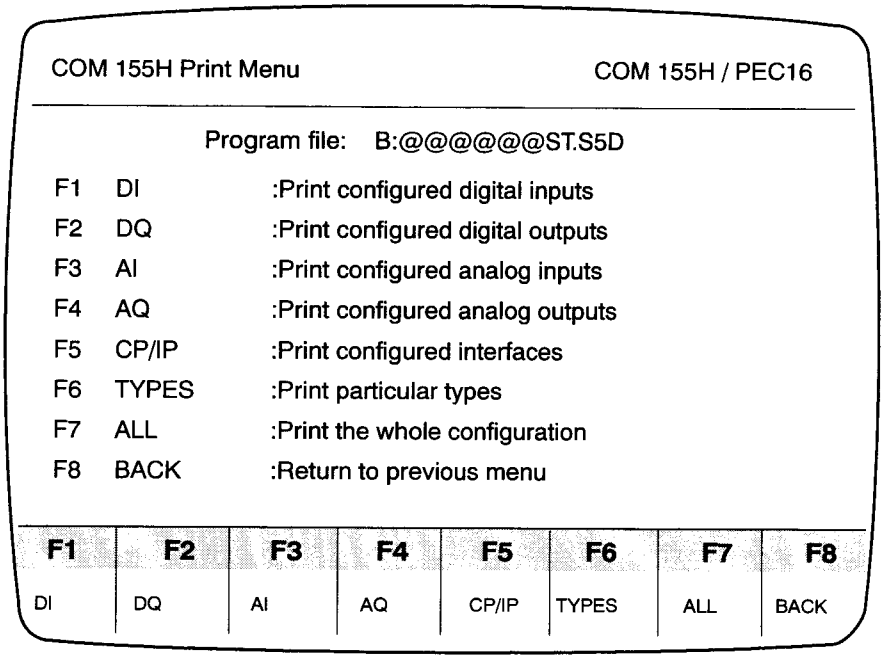


Figure 2-15 Print Menu

If the data is to be printed out from the programmable controller or diskette (FD), it must first be loaded into the programmer RAM. In the case of programmable controller and FD data, this is done via the System Handling menu, F3 <TRAN/LOAD>, and in the case of EPROM data it is done via the S5 command interpreter.

Please refer to Section 5.3 for more information on the Print menu.

F5 <PRG FILE> (in the System Handling menu):

This function enables you to create a specific program file. This setting is absolutely mandatory for all accesses (load, transfer, delete) to the hard disk or diskette. However, the function does not enable you to create a new program file. This is done with the AUX FCT (<F7>) function in the System Handling menu or in the Defaults form.

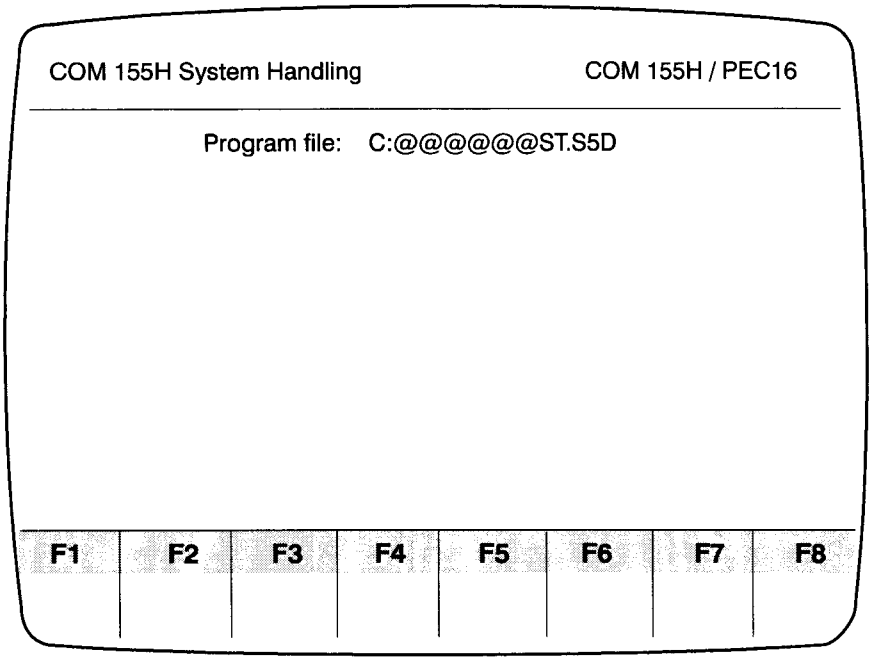


Figure 2-16 Select Program File Screen Form

Configuring and Initializing

3

From the COM 155H Basic Operating System Menu (see Section 3.1) you can reach the individual submenus via the function keys F1, F2, F3 and F4. In the submenus, you can

- Define the characteristics of the 155H operating system
- Define the DB and DX data blocks to be transferred when activating the standby
- Define the I/O areas of the individual expansion units of the switched I/O

3.1 Configuring the S5-155H PLC (S5-DOS/Stage 6)

Configuration Functions Screen Form

In this screen form, you can decide whether you want to initialize the operating system or configure the I/O.

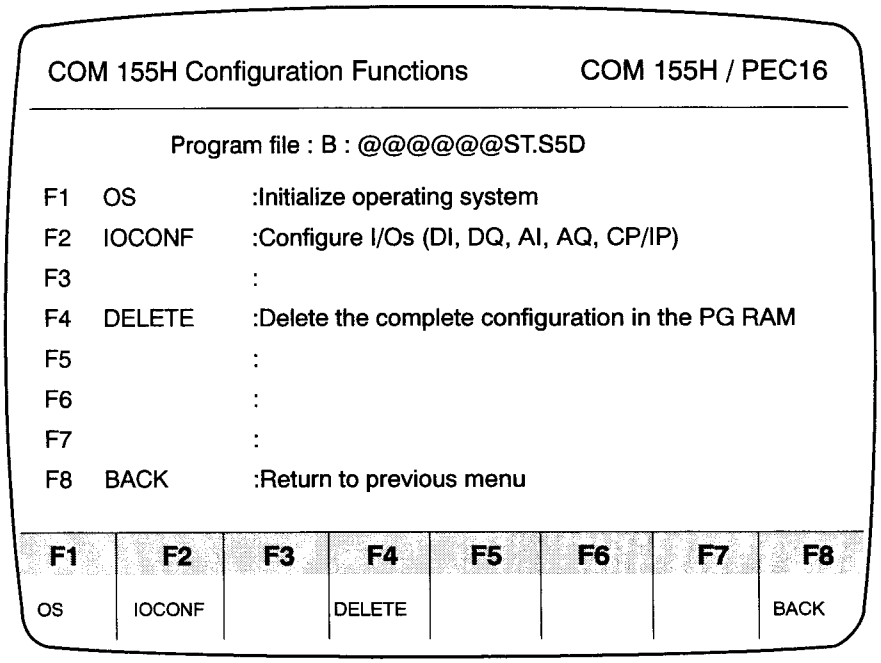


Figure 3-1 Configuration Functions Screen Form

Initializing the Operating System: <OS>

By pressing F1 in the Configuration screen form, you reach the "Initialize Operating System" menu. Press <F1> to <F3> to select the various submenus and <F8> to exit these submenus again.

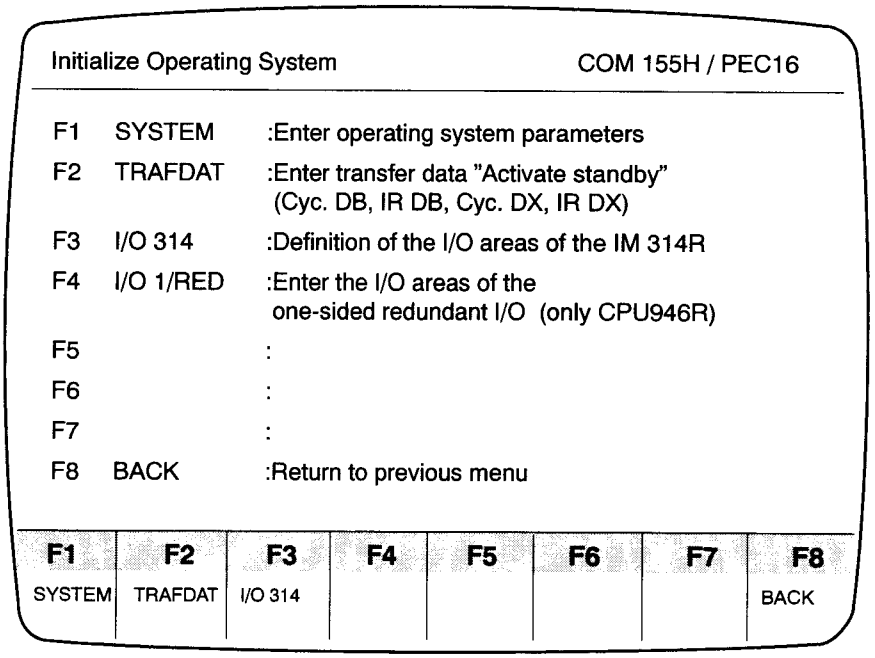


Figure 3-2 Initialize Operating System Screen Form

F1 <SYSTEM>: Initialize the redundant operating system

F2 <TRAFDAT>: Transfer data for once-only updating of the standby while activating the standby

F3 <I/O 314>: I/O areas of the expansion units of the switched I/O (IM 314R)

Initialization of the operating system is described in Section 3.2.

3

Configuring the I/Os: <IOCONF>

If you press F2 in the COM 155H Configuration Functions screen form, the basic screen form for configuring the I/Os appears on the programmer. Select one of the keys F1 to F5 to enter your configuration data for the digital and analog inputs and outputs and the CPs/IPs.

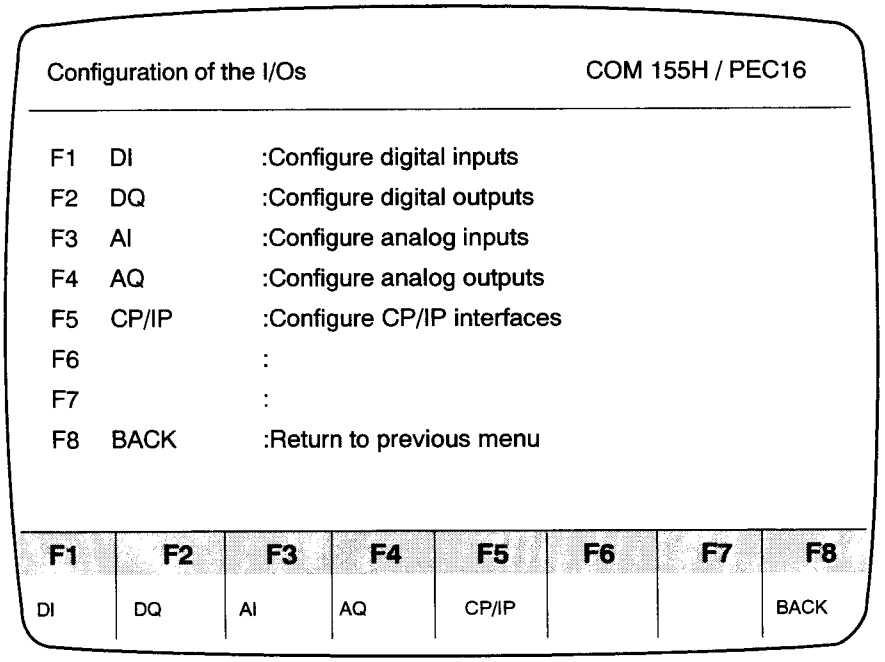


Figure 3-3 Configuration of the I/Os Screen Form

Initialization of the operating system is described in Section 3.2.

3.2 Initializing the Operating System

F1: <SYSTEM> (in the Initialize Operating System Basic Screen Form)

Operating System Parameters

You enter the relevant data for the H operating system in this screen form. If you have not loaded a DX 1 in the programmer memory (see F7 <AUX FCT> in the System Handling menu), the data input fields have default values that you can accept or modify as required.

If DX 1 has already been loaded into the programmer memory, the values initialized appear in the input fields. These can also be modified if necessary.

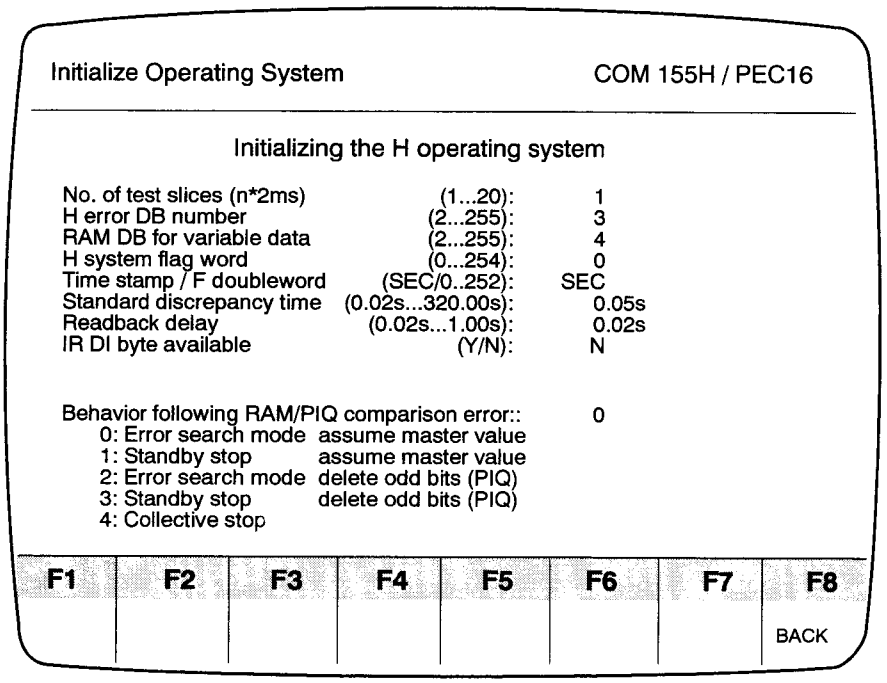


Figure 3-4 Initialize Operating System Screen Form

RAM/PIQ Comparison Error

In the case of "RAM or PIQ comparison errors", you can configure the following five operating system responses:

1. Master continues to operate unchanged and the standby enters the error search mode (default).
2. Master continues to operate unchanged, standby enters the STOP mode.
3. In the case of PIQ comparison error, the master sets the non-identical bit to 0, and the standby enters the error search mode;
   In the case of a RAM comparison error, the master leaves the non-identical bit unchanged, and the standby enters the error search mode.

4. In the case of a PIQ comparison error, the master sets the non-identical bit to 0, and the standby enters the STOP mode;
   In the case of a RAM comparison error, the master leaves the non-identical bit unchanged, and the standby enters the STOP mode.
5. Both subunits enter the STOP mode.

If a RAM or PIQ comparison error has been caused by a "stuck at 0/1" error, the self-test locates the defective side immediately and sets it to STOP with an error message. Otherwise, the operating system reacts according to the configuration.

- Position the cursor at the input field whose value you wish to change, enter the new value there and confirm by pressing <INSERT> or <RETURN>.
- Use only permissible values; these are given in parentheses. If COM 155H detects a wrong input, the error message "INVALID PARAMETER(S)" appears. Now enter a correct value or confirm the value suggested by COM 155H.

Operating System: Explanation of Parameters

Number of test slices:

Here you must enter the number of test slices that are to be used **within a PLC scan cycle** for processing the self-test routines. You can enter any number between 1 and 19.

Processing of a test slice takes 2 ms. The more test slices you process, the shorter the error detection time will be; however, the scan cycle time is increased accordingly.

H error DB number (number of error data block in the H system):

This field is for a DB number anywhere between 3 and 255. The H operating system needs this data block to manage H system errors. You can read out the data block with COM 155H support (see Section 5.2).

If you enter a number that has already been used for the RAM data block, the message

"DB NUMBER ASSIGNED"

will appear.

RAM DB for variable data

Here you can also enter any data block number between 3 and 255. The system program requires this data block to manage redundancy-specific values. It must be in the RAM.

The length of this data block depends on the scope of your system configuration and varies between one word and 32 Kwords.

If you enter a number already used for the H error data block, the message

"DB NUMBER ASSIGNED"

will appear.

H system flag word

Here you can enter a flag word between 2 and 254. One byte of this word is reserved for redundancy-specific status information entered by the H operating system. The second byte is reserved for control information that can be set by the user in his STEP 5 program.

Please refer to Section 8.5 of the S5-155H Programmable Controller Instructions for information on the structure of the H system flag word. If the value you have entered coincides with the value entered for the flag doubleword, the entry will be rejected with the message

"DOUBLE FLAG ASSIGNMENT"

Time stamp F doubleword

If you specify SEC (time stamp: year month day hour minute second) here as the parameter, the CPU real-time clock (with data and time of day) is automatically entered in the time stamp of the H error DB in the event of an error.

If you specify any flag doubleword between 0 and 255, the information in this doubleword is supplied to the time stamp in the H error DB automatically if an error message is entered there.

You can store the identifiers (for example, cycle counter, sequencer, etc.) in this flag doubleword.

Please refer to Section 8.3 of the S5-155H Programmable Controller Instructions for information on the H flag doubleword.

If the value you enter coincides with the value entered for the H system flag word, your entry will be rejected with the message

"DOUBLE FLAG ASSIGNMENT"

Standard discrepancy time

Redundant digital inputs and redundant analog inputs may have different signal states or input values over a relatively short time. By specifying a "discrepancy time", you can determine how long the H system is to tolerate such different signal states. A discrepancy time of between 0.01 and 320 s is permissible.

The standard discrepancy time specified in the Initialize Operating System screen form is displayed as default in the Configuration of the I/Os screen forms.

- Proceed as follows when entering the time:
  - Enter the time value before the decimal point
  - Press <RETURN>
  - Enter the time value after the decimal point
  - Press <RETURN>

IMPORTANT
This procedure applies to all time entries in COM 155H!

Readback delay

The various digital output modules have different signal propagation delays. For this reason, you must specify here the time by which the reading in of the readback DIs (digital inputs) is to be delayed.

This time then applies to **all** redundant digital outputs (see Readback delay times table in Section 11.3 of the S5-155H Instructions).

- When entering the time value, please proceed as described above.

IR DI byte available

Here you specify whether the input byte IB 0 (process-interrupt-driven program processing) is to be used as an interrupt DI byte or not.

Note:
IB 0 can only be operated as an interrupt DI in switched or redundant mode!

The maximum discrepancy time is 1.00 s. The discrepancy time specified for bit 0.0 applies to all eight inputs.

3.3 Initializing Activation of the Standby
F2: <TRAFDAT> (in the Initialize Operating System Basic Screen Form)

Transfer Data for Activating Standby

Use this form to enter all the data blocks that are to be transferred from the master to the standby when the latter is activated. These are the DB and DX data blocks.

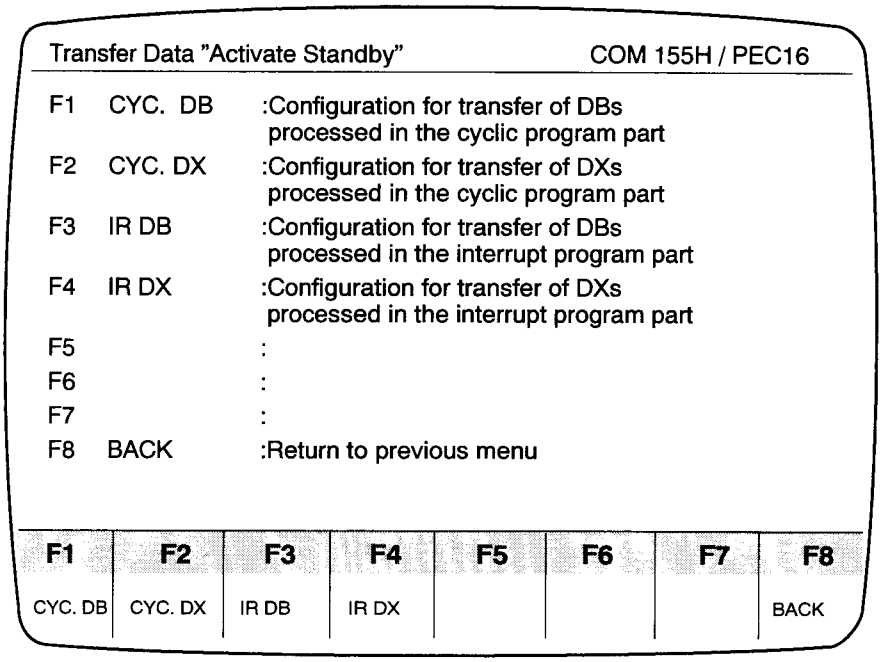


Figure 3-5 Transfer Data Screen Form

The contents of the DB and DX data blocks can be modified by the user program. A distinction is made between data blocks processed in the cyclic program (for example, DB or DX from OB 1 and the blocks called from there), and data blocks processed in timed interrupts and process interrupts (for example DB or DX from OB 13, timed interrupt).

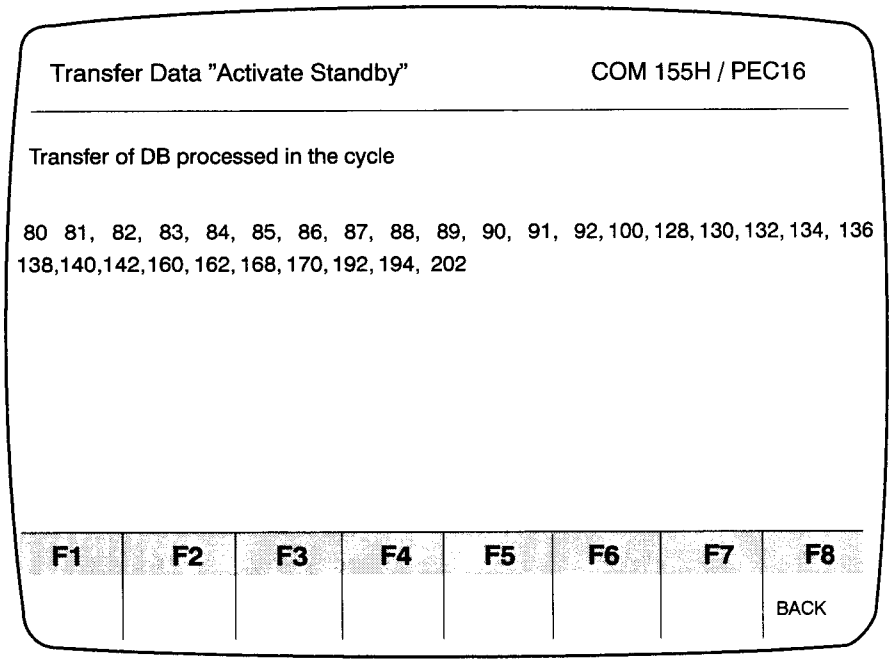
Data blocks that occur on both levels need only be entered in the interrupt DB/DX screen form.

The more data blocks you enter in these four screen forms, the longer the once-only update of the standby in the activation phase will take.

The more data blocks that have been entered in the interrupt DB/DX, the longer the interrupts are disabled by updating.

If the H error DB for the RAM DB is in one of the lists, it is automatically deleted from the list by COM 155H.

Example



Only DB/DX numbers between 3 and 255 are permissible. If a wrong number is specified, the message

"NUMERIC VALUE ILLEGAL"

appears.

- If you press <RETURN> repeatedly, COM 155H automatically increments the numerical value.

  Example:
  If you enter "5" and press <RETURN> four times, the numerical sequence 5, 6, 7, 8, 9 appears on the screen.

- If you have to enter a longer numerical sequence, you can use the following convenient method:

  Key in "30" "–" "75", for example, and COM 155H automatically generates the complete numerical sequence from 30 to 75.

- If you wish to delete individual DB/DX numbers, overwrite these with the <space bar>.

- When you have entered all data blocks, press F8 <EXIT>. The message "Data sorted and accepted"

  appears.

- Any entries made can be selected with the cursor.

3.4 Initializing the I/O Areas

F3: <I/O 314> (in the Initialize Operating System Basic Screen Form)

Area Parameters of the Switched I/O in the EU

This screen form is used to define the I/O areas of the IM 314R (switched I/O). The area numbers specified in the IM 314R I/O form automatically assign the relevant I/O area to the IM 314R and, consequently, to the associated expansion units.

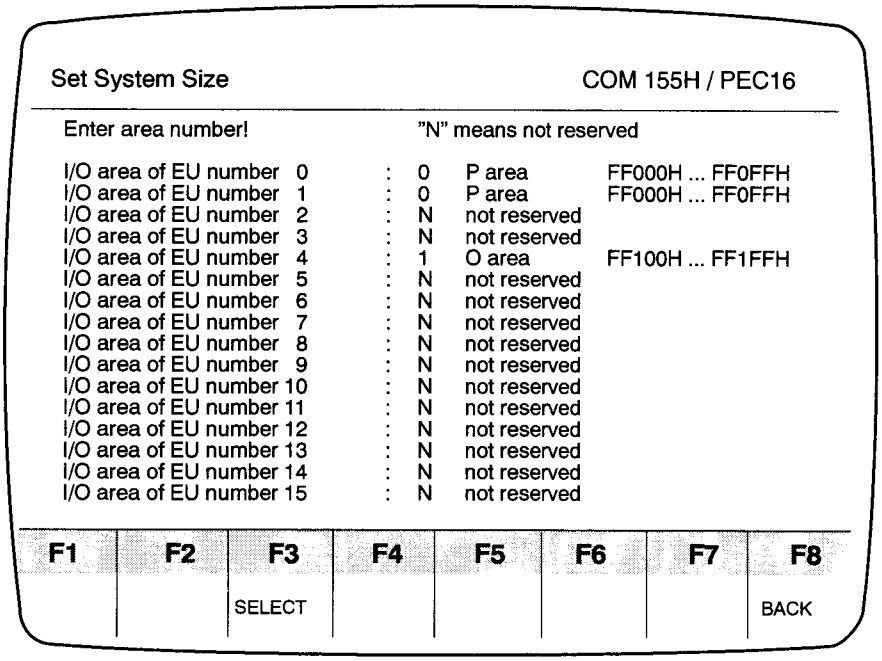


Figure 3-6 IM 314R I/O Screen Form

Only block numbers 0, 1, 3, 12 and 13 are permissible. If you specify a wrong number, the message

"INVALID PARAMETER(S)"

appears.

When you press F8 <EXIT>, the message

"Data accepted"

appears to confirm this.

Configuring the I/O

4

The F1 to F5 keys in the I/O basic screen form take you into the individual I/O forms in which you can enter your configuration data for the digital and analog inputs/outputs, as well as the CPs and IPs (see Section 4.2).

4.1 General Structure of the I/O Configuration Screen Forms

I/O Basic Screen Form

The I/O Configuration screen form in COM 155H is structured according to the following pattern:

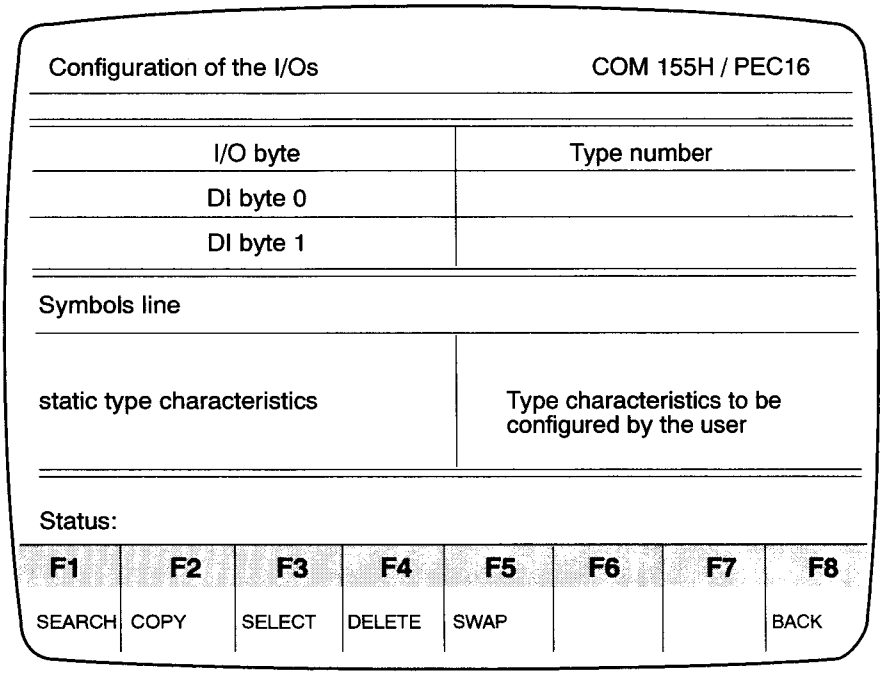


Figure 4-1 Structure of the COM 155H I/O Configuration Screen Form

To help you configure your I/O bytes/words, the S5-155H has a number of different I/O types. By specifying a **type number** for a specific I/O byte, you define the following:

a) the signal type: DI, DQ, AI, AQ, CP/IP

b) the mode: one-sided, switched, redundant, three-channel redundant

Address area:

DI	0 ... 255	P area
DI	256 ... 511	O area
DQ	0 ... 255	P area
DQ	256 ... 511	O area
AI	128 ... 254	P area
AQ	256 ... 510	O area

The following table contains all configurable I/O types

Table 2-1 Configurable I/O Types

Type No.	Meaning	Fault Tolerance
1	DI byte 1-channel	Standard (as for S5-155U)
2	DI byte switched	Enhanced
3	DI byte 2-channel redundant	High
4	DI byte 3-channel redundant	Highest
8	DQ byte 1-channel	Standard
9	DQ byte switched	Enhanced
10	DQ byte 2-channel redundant	High
11	DQ byte 2-channel redundant	High, with 3 R-DIs
13	AI channel 1-channel	Standard
14	AI channel switched	Enhanced
15	AI channel 2-channel redundant	High
16	AI channel 3-channel redundant	Highest
18	AQ channel 1-channel	Standard
19	AQ channel switched	Enhanced
20	AQ channel redundant	High (without error location)
21	AQ channel redundant with error location	High
24	CP/IP 1-channel	Standard
25	CP/IP switched	Enhanced

Displays in the Configuration Screen Form

In the top left corner of the Configuration of the I/Os screen form (see Fig. 4-1), the I/O byte/word or the interface number is displayed and, to the right of this, the associated type. The next I/O bytes/words follow underneath.

When a specific configuration screen form (DI, DQ, AI etc.) is selected, the cursor is located in the Type number field.

- If you press F3, the lowest associated type number in each case is displayed ("1" for DI, "8" for DQ, etc.)
- You can select another type by pressing F3 <SELECT> (e.g. for DI ring selection 1, 2, 3, 4, 1, 2, 3, 4, 1, etc.)

The **symbols** line indicates the I/O byte or word the cursor is currently positioned at (for example, digital input 2.3 or analog output 128). In addition, the associated abbreviated symbol (eight characters) and the non-abbreviated symbol (14 characters) are indicated here, provided you have created them (for the digital I/O area 0 to 127).

Example:

Symbols line: Valve 1 Valves for pumps 0 to 7

- Confirm the type by pressing <RETURN> or <INSERT>. The associated characteristics field appears. The cursor is located in the right-hand half of this field.

Processing the I/O Configuration Screen Form

The bottom half of the Configuration form contains the **characteristics field** in which the characteristics of the current type are displayed. The left-hand field contains the fixed or static characteristics assigned to a particular type. The right-hand field contains the characteristics to be configured by the user for the current type.

- Enter the necessary data and confirm each entry with <RETURN> or <INSERT>.

When you have made your last entry, the cursor will jump back up to the line of the next byte/word.

- The desired byte or word number can be selected with the <Cursor up> and <Cursor down> keys (scroll function).

When you have configured your system and stored it in the programmer memory, bytes/words that have already been configured will be automatically displayed when you make entries. This applies also to inputs or output already reserved as L-DI, L-DQ and R-DI. You can then no longer specify a type number.

The current processing status ("TYPE INPUT" or "SWAP", for instance) is displayed in the left-hand half of the **status and error line**; error messages appear in the right-hand half.

Keys in the Configuration Screen Form

F1: <SEARCH>

This function enables you to select a random byte, word or interface number quickly and without having to use the cursor.

- Press <F1> and enter the byte/word or interface number you are looking for. The cursor then appears in the line of the specified byte/word or the specified interface number.

F2: <COPY>

This function allows you to copy the configuration of a particular byte/word or interface number to another byte/word or another interface number.

After you have entered the desired byte number(s), the message

"DESTINATION END AT BYTE x COPY?"

appears on the screen.

Confirm with the <INSERT> key.

If you make an illegal specification, COM 155H aborts and an error message appears, for example:

"DESTIN. AREA IN SOURCE AREA" or

"READBACK I/O: ABORT".

The following cannot be copied:

- L-DI
- L-DQ
- R-DI
- DI type 4
- DQ types 10 and 11

- AI type 16
- AQ type 21

F3: <SELECT>

When the cursor is in the top-right Type number field, you can use this specification to select all possible I/O types (for example, DI ring selection 1, 2, 3, 4, 1, 2, 3, 4, 1, etc.). Confirm the type number displayed with the <RETURN> key. Further, you can use the <SELECT> key for making selections in the characteristics field (for instance, AQ type 21).

F4: <DELETE>

This function deletes one or more bytes/words in your configuration. When you have entered the byte number(s), the question "DELETE?" appears on the screen.

- Confirm with the <INSERT> key or press <ESC>.
- If you want to delete a number of successive bytes, you can specify, for example, "10" "-" "15". COM 155H then deletes bytes 10, 11, 12, 13, 14 and 15.

F5: <SWAP>

This function swaps configuration data between individual bytes words.

The following message appears on the screen:
"DESTINATION END IN BYTE x ABORT?"

- Confirm with <INSERT> or press <ESC>.

The following cannot be swapped:

- L-DI
- L-DQ
- R-DI

} COM 155H generates the error message "READBACK I/O: ABORT"

- DI types 1+2 in the analog area (AI and AQ screen forms, 128 to 254)
- DQ types 8+9 in the analog area (AI and AQ screen forms, 128 to 254)

4.2 Structure of the Individual I/O Configuration Screen Forms

When configuring your I/Os, please refer to the relevant chapter in the S5-155H Instructions.

Digital Inputs: F1 **<DI>** (in the I/O basic screen form)

The COM 155H Configuration of the I/Os Type 1 form appears on the screen.

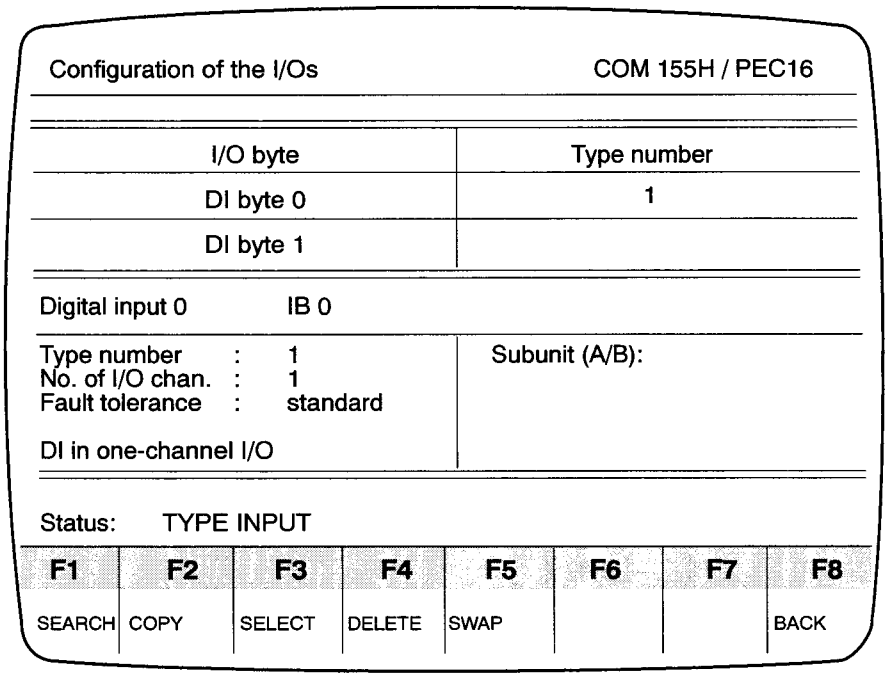


Figure 4-2 I/O Configuration Screen Form Type 1

All you have to do here is program the subunit in which the DI is operated.

Using <F3>, you can preselect the individual type numbers. These are shown on the screen in plaintext. This is valid for the entire configuration of the I/O.

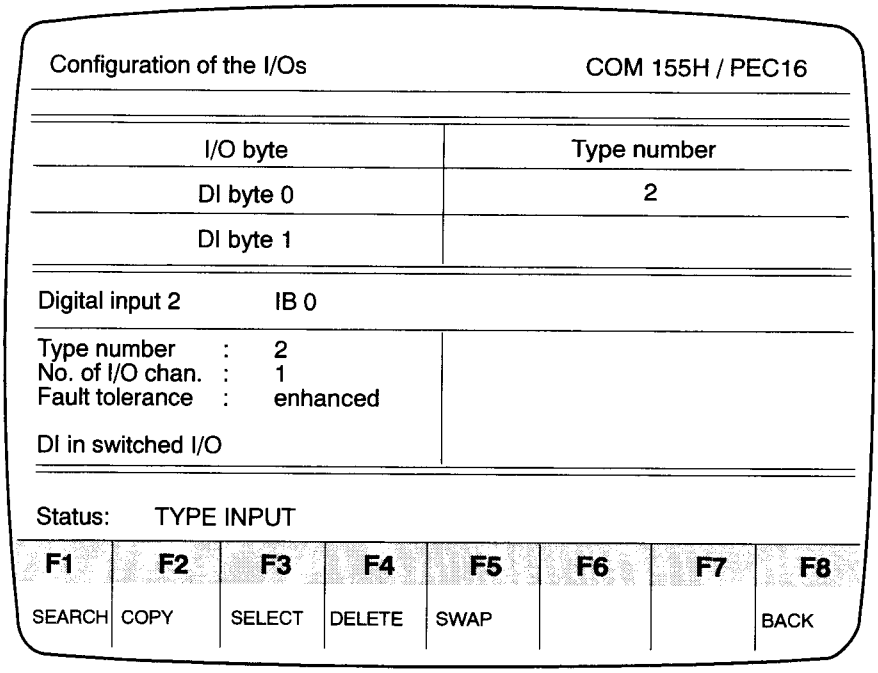


Figure 4-3 I/O Configuration Screen Form Type 2

No further initialization is necessary here.

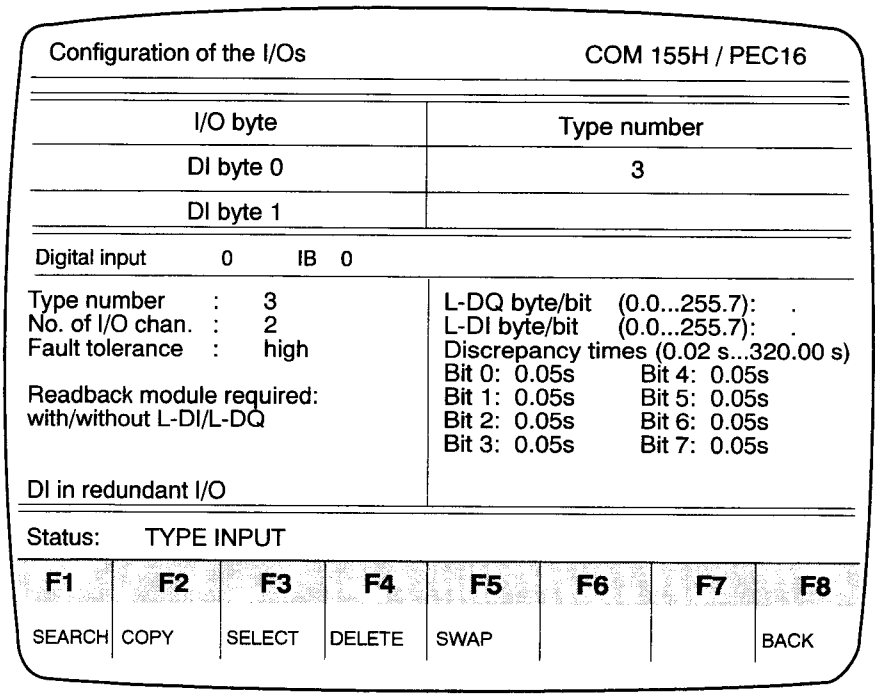


Figure 4-4 I/O Configuration Screen Form Type 3

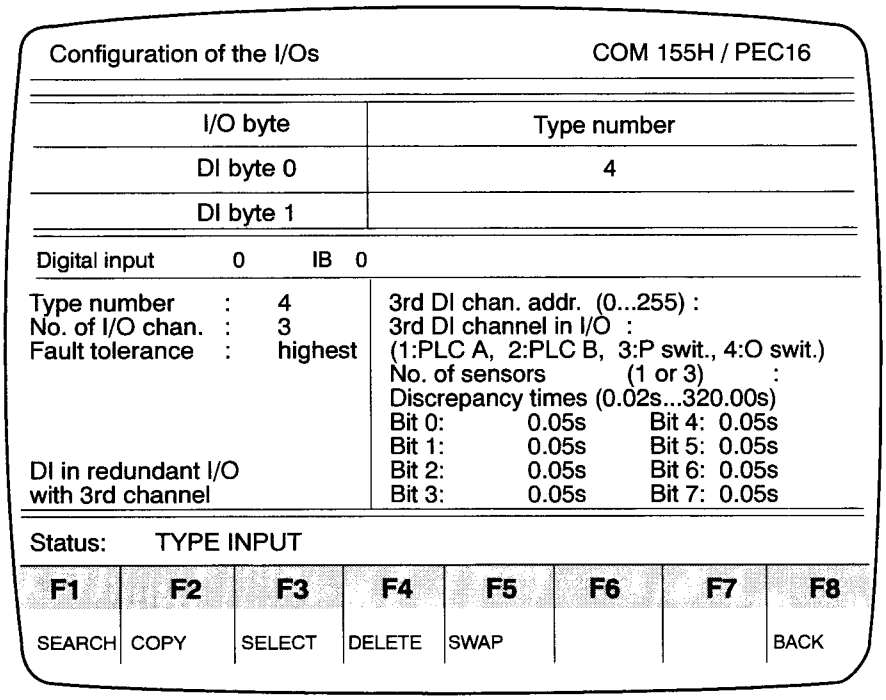


Figure 4-5 I/O Configuration Screen Form Type 4

Legend: DI

Locating DI, locating DQ:
You can use type 3 with or without error locating DI (L-DI) or L-DQ: If you use L-DI or L-DQ, the DI can be used as "NON-STOP DI".

Note:
Several redundant DIs can use the same locating facility (L-DI/L-DQ).

Discrepancy times
Since redundant digital inputs may have different signal states over a comparatively short period of time, you can use COM 155H to configure how long these different signal states can be tolerated.

If the configured discrepancy time is smaller than a PLC cycle time, the discrepancy time is set within the CPU to a PLC cycle time during cyclic processing (apart from with redundant process interrupts).

Various discrepancy times can be assigned to the individual bits of the DI. You can configure discrepancy times of between 10 ms and 320 ms in steps of 10 ms. The configured standard discrepancy time in the COM 155H "Initialize Operating System" screen form is the default value.

Digital Outputs: F2 **<DQ>** (in the I/O basic screen form)

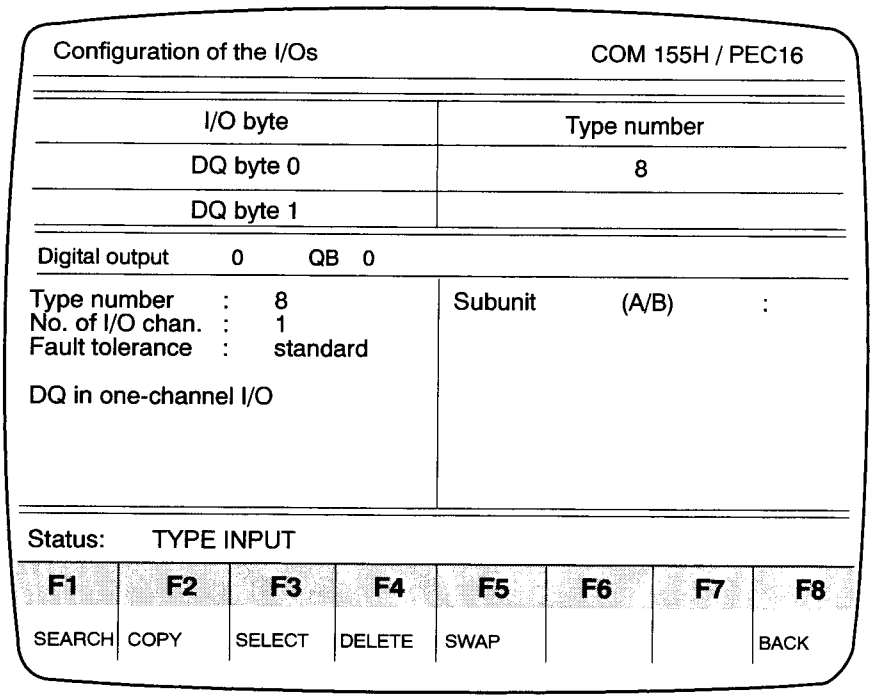


Figure 4-6 I/O Configuration Screen Form Type 8

Here you only have to program the subunit in which the DQ is operated.

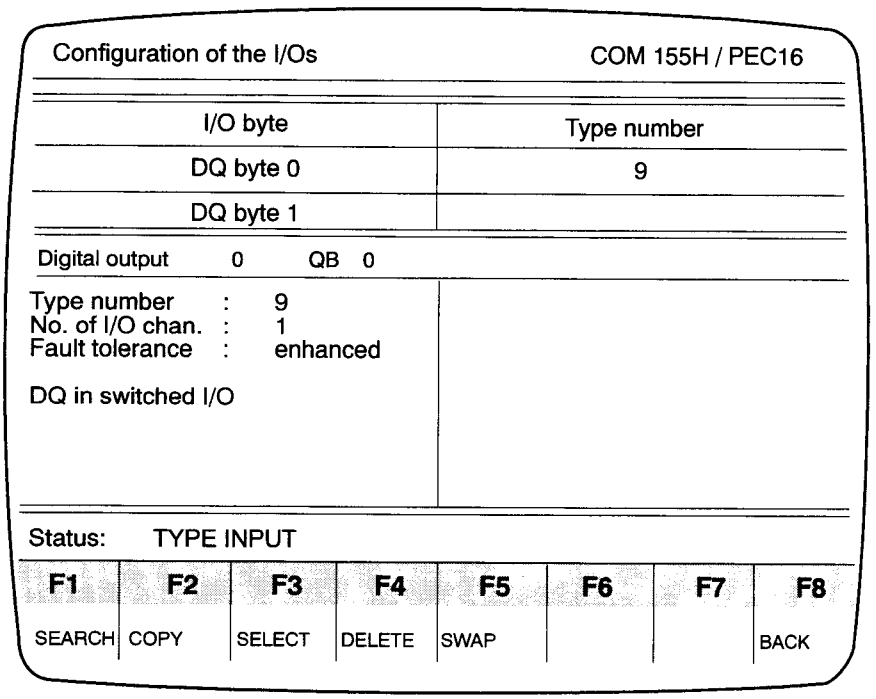


Figure 4-7 I/O Configuration Screen Form Type 9

4

No other parameters need be assigned here. However, remember to configure the I/O area of the relevant expansion unit using COM 155H.

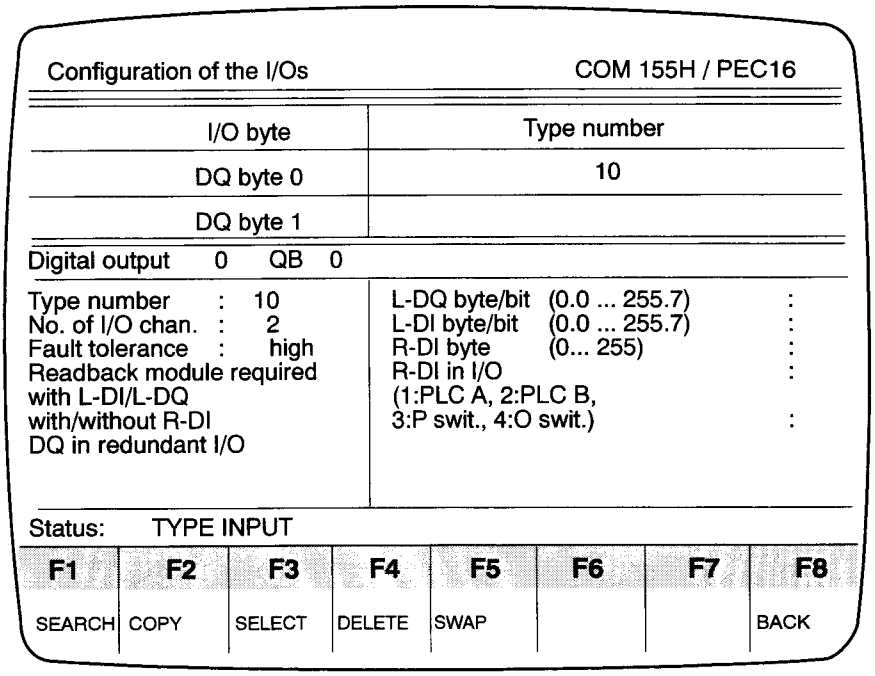


Figure 4-8 I/O Configuration Screen Form Type 10

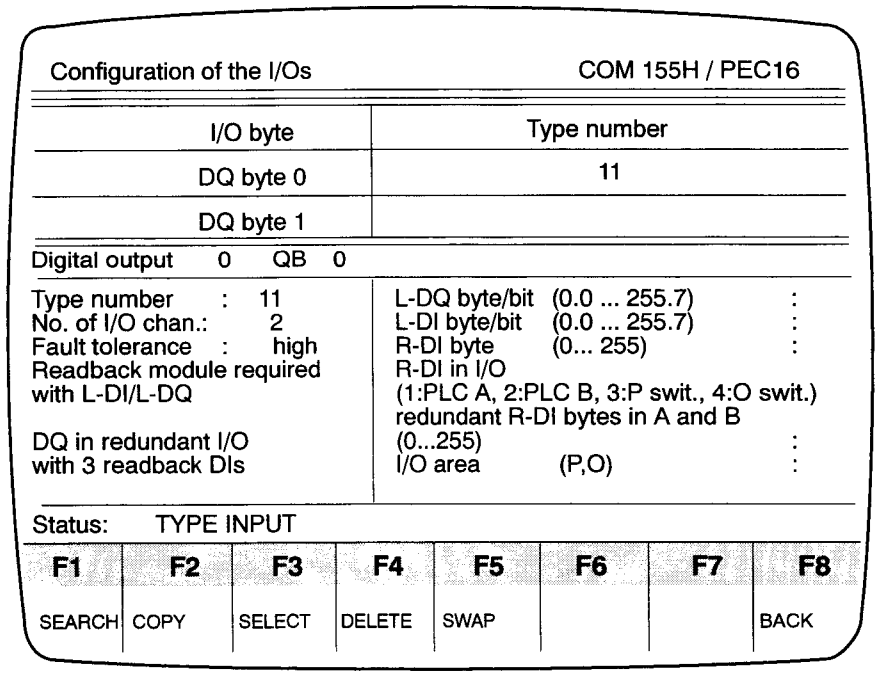


Figure 4-9 I/O Configuration Screen Form Type 11

Legend: DQ

Locating DI (L-DI), locating DQ (L-DQ):
see DI in I/O Configuration Screen Form Type 3

Readback DI (R-DI)
A readback DI must be specified for each redundant DQ, otherwise an error cannot be detected.

You must also specify a readback delay in the COM 155H "Initialize Operating System" screen form for the readback DIs. This takes the different signal propagation delays of the various digital output modules into account.

Readback DI in the I/O
In this field, you specify the type of I/O in which the readback DI is to be operated.

1. The readback DI is one-sided: Subunit A.
2. The readback DI is one-sided: Subunit B.
3. The readback DI is switched: P area
4. The readback DI is switched: O area

Redundant readback DI
For the redundant DQ type 11, the redundant readback DI bytes must also be specified.

Analog Inputs: F3

<AI> (in the I/O basic screen form)

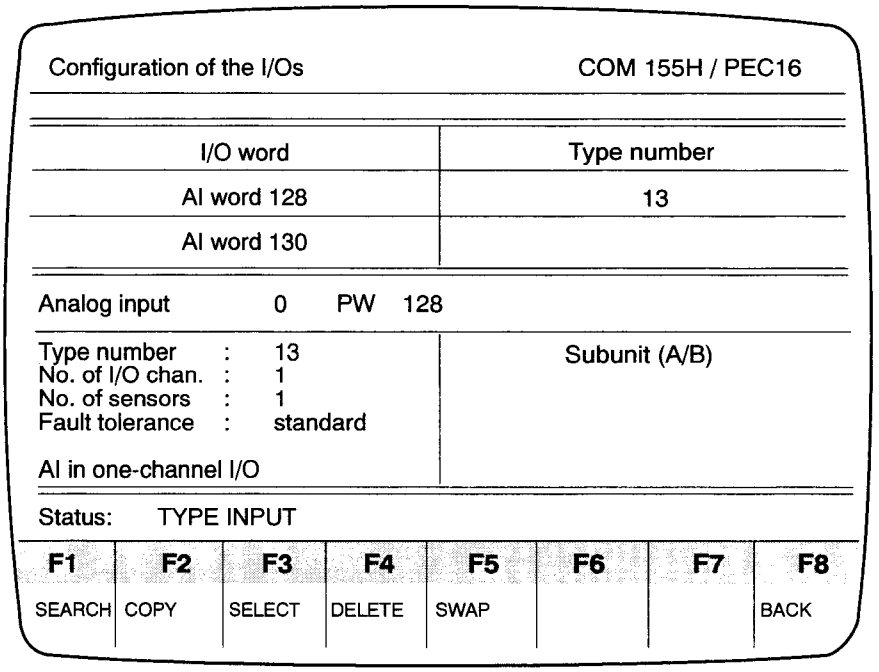


Figure 4-10 I/O Configuration Screen Form Type 13

Here you must program the subunit in which the AI is operated.

4

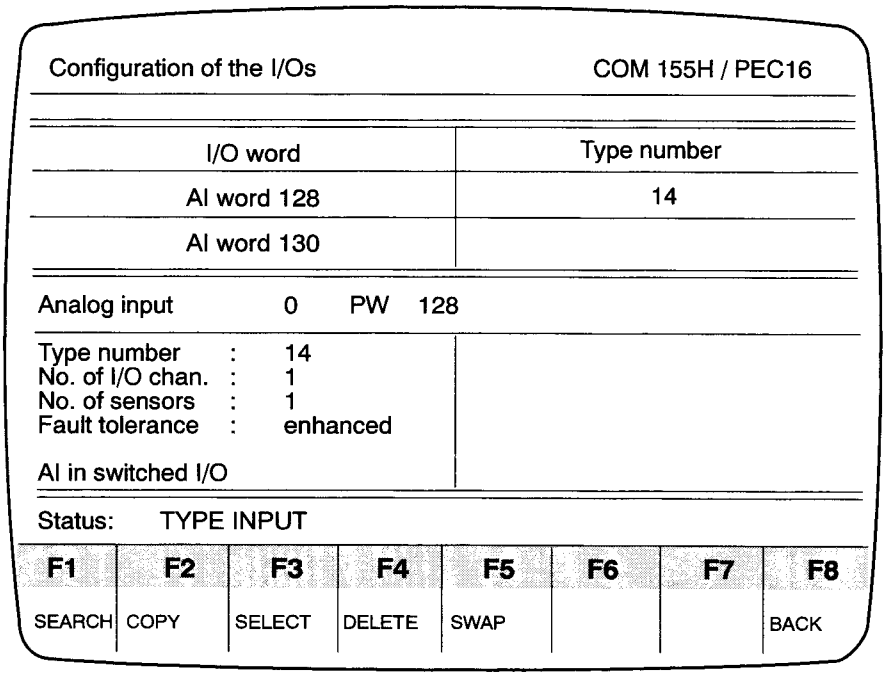


Figure 4-11 I/O Configuration Screen Form Type 14

No other parameters need be assigned.

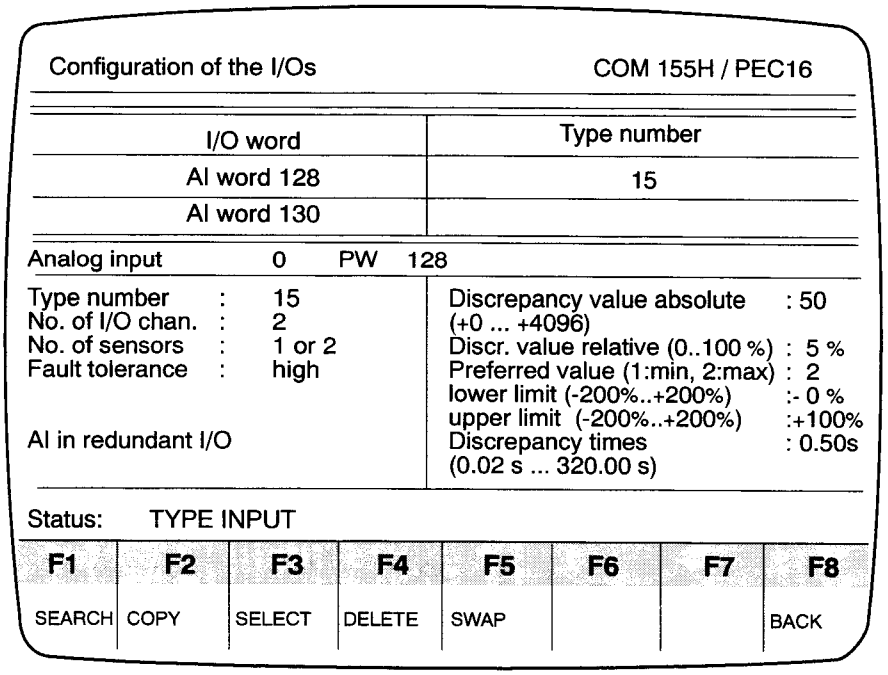


Figure 4-12 I/O Configuration Screen Form Type 15

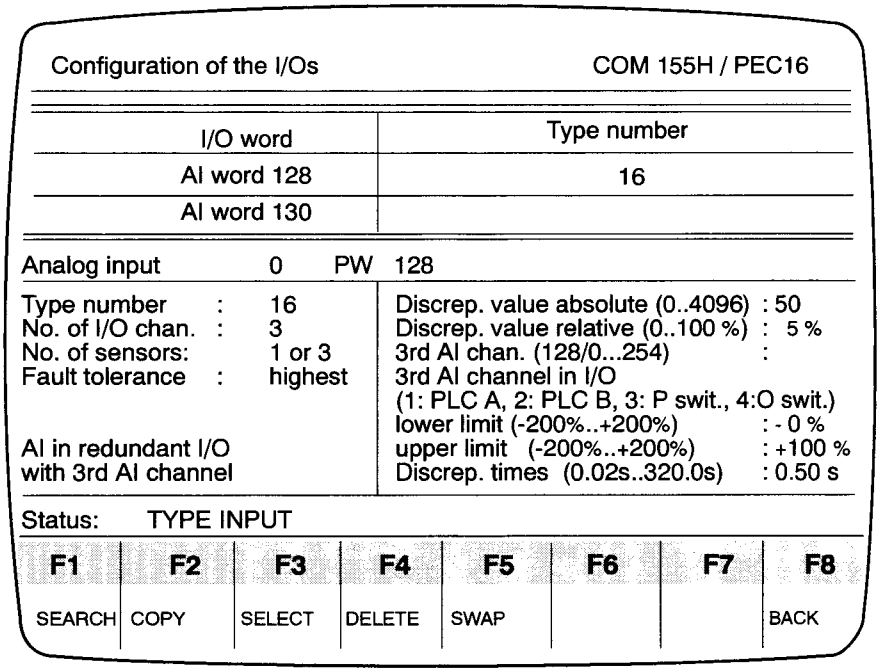


Figure 4-13 I/O Configuration Screen Form Type 16

Legend: AI (Types 15 and 16)

Absolute/relative discrepancy value:
Enter an absolute value ABS (decimal number) and a relative value REL (percentage).

The 155H operating system calculates the permissible analog value discrepancy DPERM according to the following formula:

$$D_{PERM} = ABS + \frac{REL * RAWV\ (max)}{100}$$

where for type 15 RAWV (max) is the greater of the two instantaneous analog values.

where for type 16 RAWV is the middle of the three instantaneous analog values.

Preferred value (type 15):
Specify here whether the 155H operating system is to give preference to the maximum or minimum value in the case of a discrepancy between the analog values.

Lower, upper limit (type 15/16):
The upper and lower limit values define a range for the analog value outside which the 155H operating system reports an error.

Discrepancy time (type 15/16):
See Configuration screen form DI type 3

Analog Outputs: F4

<AQ> (in the I/O basic screen form)

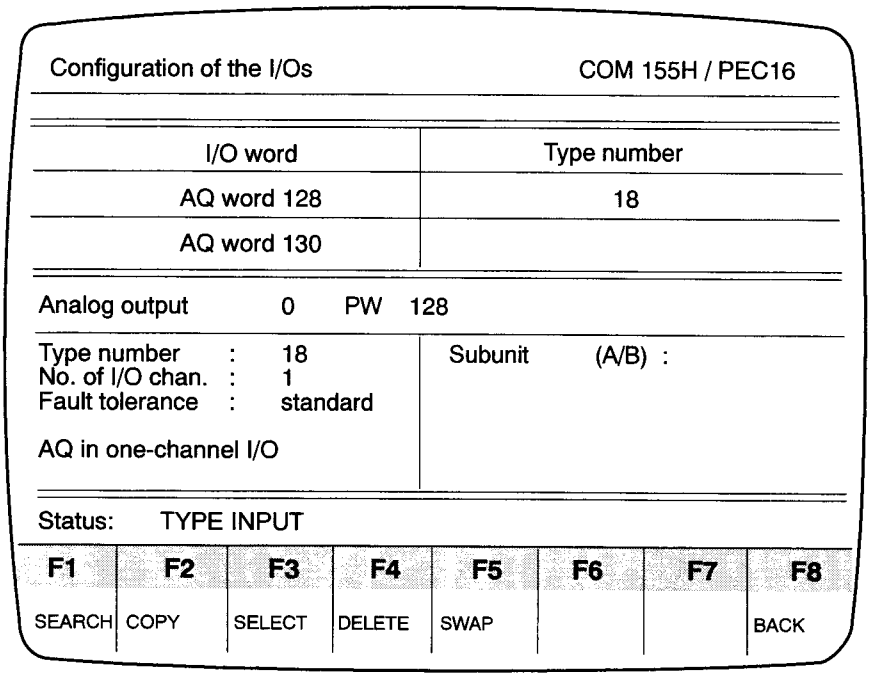


Figure 4-14 I/O Configuration Screen Form Type 18

Here you must specify the subunit in which the AQ is operated.

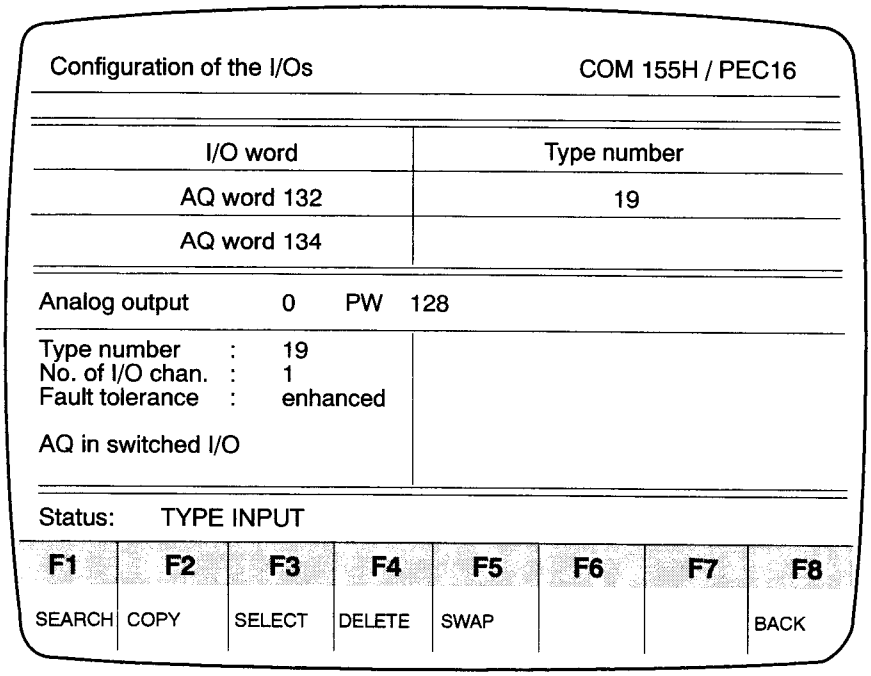


Figure 4-15 I/O Configuration Screen Form Type 19

No other parameter settings are required here.

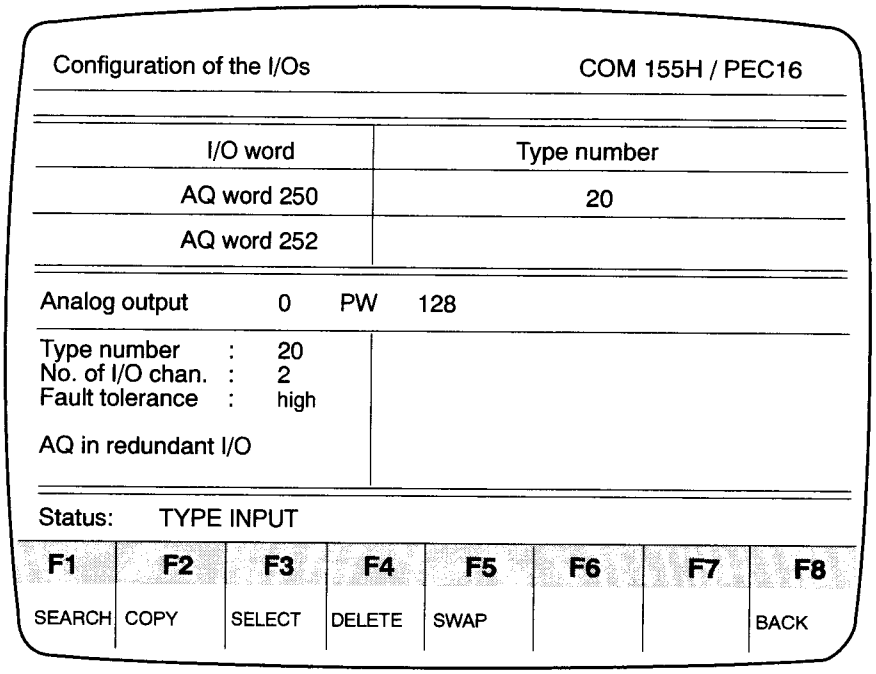


Figure 4-16 I/O Configuration Screen Form Type 20

The redundant analog output type 20 cannot locate errors.

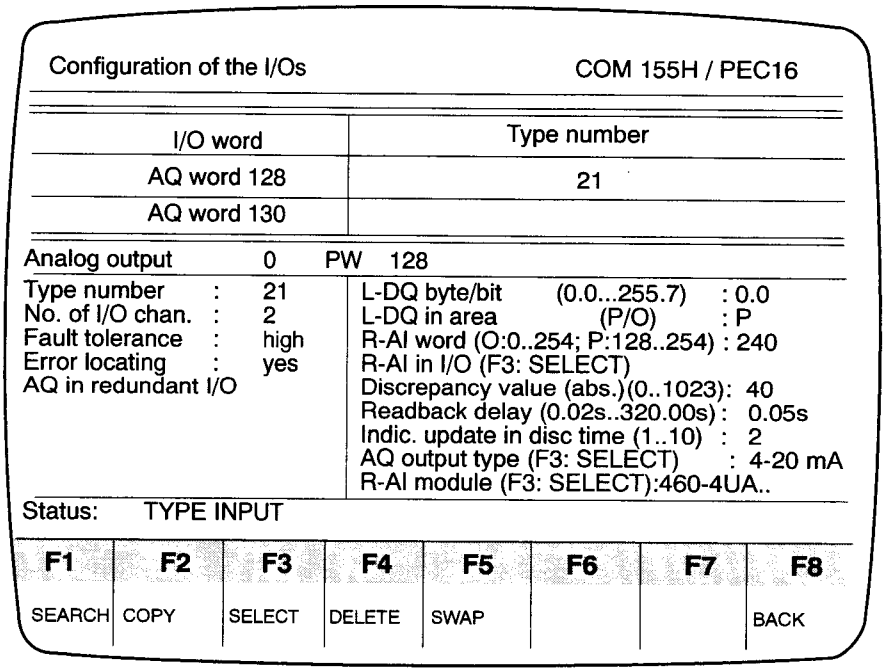


Figure 4-17 I/O Configuration Screen Form Type 21

The redundant analog output type 21 can locate errors and remedy them.

Legend: AQ Type 21

L-DQ byte/bit
Area: P or O (the L-DQ byte and bit lie in subunit A and B at the same address).

1. An L-DQ byte which is configured for a two-channel AQ may not also be used elsewhere (not even as the L-DQ for DI or DQ), because if a subunit fails, the byte on the intact side is written with 0FFh.
2. Assignment of channel number to L-DQ bit number is fixed:

   $$\frac{\text{Address AQ}}{\text{2 modulo 8}} = \text{L-DQ bit no.}$$

   Address AQ = base address + channel no. *2
   Example:
   Base address = 128 and channel no. = 5:
   → address AQ = 138, L-DQ bit no. = 5.
   Base address = 200 and channel no. = 3:
   → address AQ = 206, L-DQ bit no. = 7.

L-DQ Bit	Address of AQ Channel							
0	128	144	160	176	192	208	224	240
1	130	146	162	178	194	210	226	242
2	132	148	164	180	196	212	228	244
3	134	150	166	182	198	214	230	246
4	136	152	168	184	200	216	232	248
5	138	154	170	186	202	218	234	250
6	140	156	172	188	204	220	236	252
7	142	158	174	190	206	222	238	254

3. Free L-DQ bits may not be used elsewhere.

R-AI address
Area: P or O I/O area: one-sided in A or B or switched.

(Permissible) discrepancy value (absolute)
Suggestion for systems with minimal failure rate: 40

Readback delay
Default: 0.05 s
Readback delay time to be configured = R-AI encoding time or
R-AI cycle time + possibly ET 200 bus cycle time.

Indic. no. of updates (AQ) (FB calls and transfer direct access T PW/T OW) **in discrepancy time (AQ)** (1...10)
Default: 10
The value for "Indic. update in disc. time" is:
N = readback delay time/AQ update interval.
Example 1 for "Indic. update in disc. time" = N

- AI 463: R-AI encoding time = 50 ms
- AQ updates every 10 ms (FB calls or T PW instructions on AQ)
  → N = 50/10 = 5

Conditions

U Periphery:

The FB 41 : H-RLG : AA can be used for the following modules:

- AQ modules:
  6ES5 470-4UA..
  6ES5 470-4UB
  6ES5 470-4UC
- Readback AI modules:
  6ES5 466-3LA.. encoding time < 4 ms

  As this module has no interference suppression it can only be used as an R-AI if the high-frequency interferences arising on the user side are small enough.

  If the module is used as an R-AI for current outputs, only the 8 differential inputs may be used.
  6ES5 460-4UA.. encoding time < 480 (960) ms
  6ES5 463-4U... encoding time < 50 ms,

  only as R-AI for voltage output
  6ES5 465-4UA.. encoding time < 480 (960) ms,
- L-DQ modules:
  6ES5 458-4UA..

CP/IP: F5

<CP/IP> (in the I/O basic screen form)

Here you must specify the subunit to which communications processors/intelligent I/O modules have been assigned.

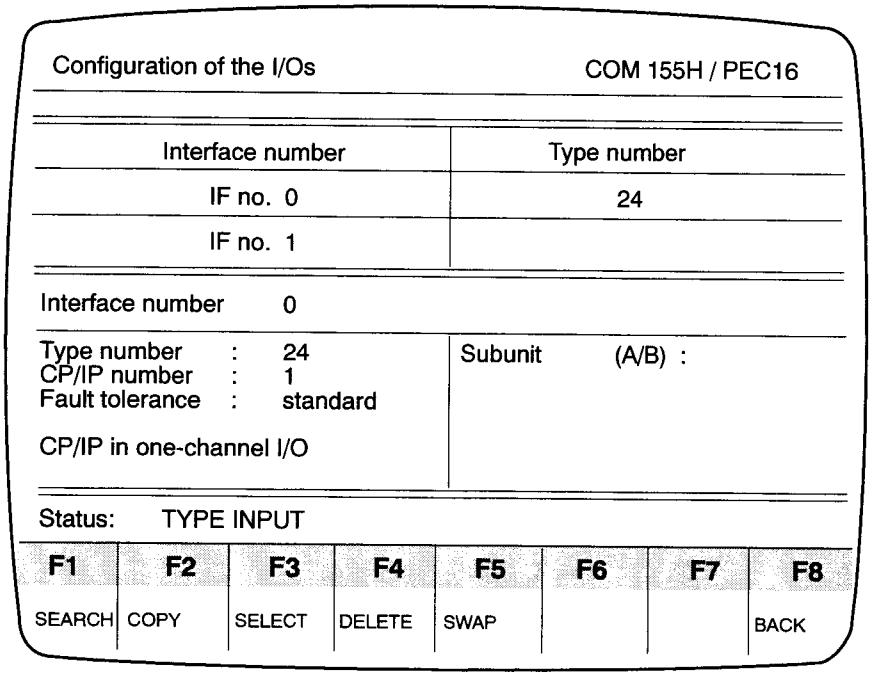


Figure 4-18 I/O Configuration Screen Form Type 24

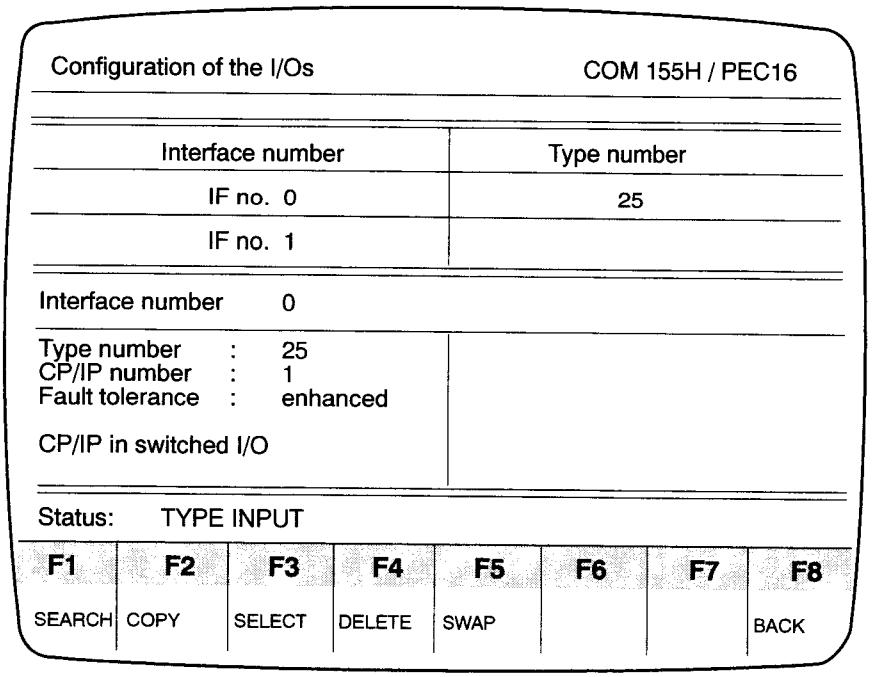


Figure 4-19 I/O Configuration Screen Form Type 25

No other parameter settings are required here.

Error Diagnostics and Documentation 5

The F2 and F4 function keys in the Diagnostics basic form support you when troubleshooting. You can obtain an overview of the errors that have occurred with F2 <STAT ERR>, as well as detailed information on each individual error with F4 <H ERROR>.

If you want to document your configuration, press <F7> in the Main Menu to reach the "System Handling" menu and then press <F4> to reach the "COM 155H Print Menu".

5.1 Static Error Image of the I/Os

I/O Error Image Display

F2 <STAT ERR> in the Diagnostics menu takes you to the basic screen form for the I/O error image. All messages entered in the STATUS word (= DW 3 of the error DB) appear. If, for example, the "CONFIGURATION ERROR" message appears, there is at least one and possibly several configuration errors (please refer to the S5-155H Instructions, Section 8.2 "Structure of the Error DB").

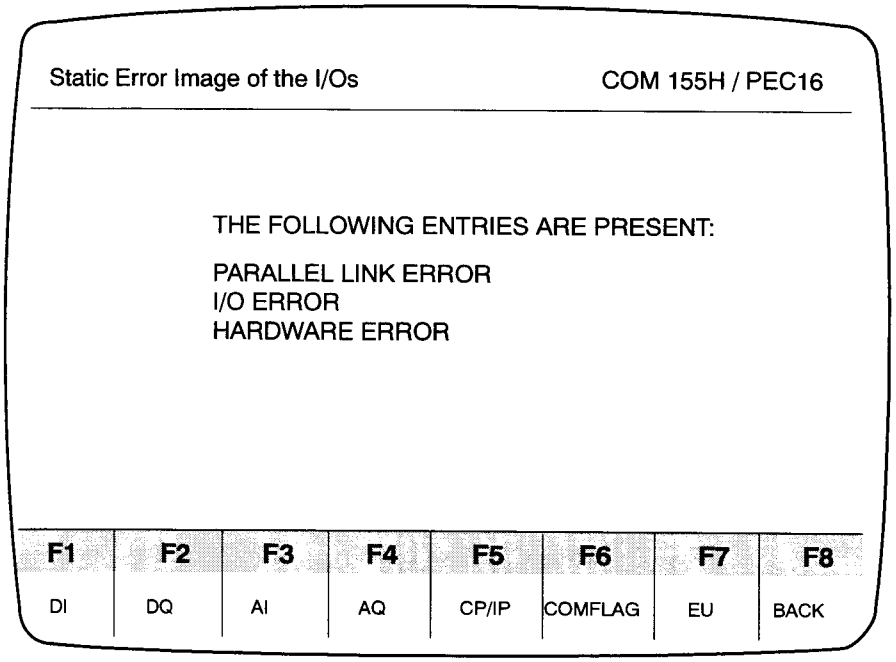


Figure 5-1 Basic Screen Form of the I/O Error Image

You can then have the static error image displayed as follows:

- Digital inputs <DI>
- Digital outputs <DQ>
- Analog inputs <AI>
- Analog outputs <AQ>
- CP and IP modules <CP/IP>
- Interprocessor communication input and output flags <COMFLAG>.

If you press F1 <DI>, for example, the following screen form will appear:

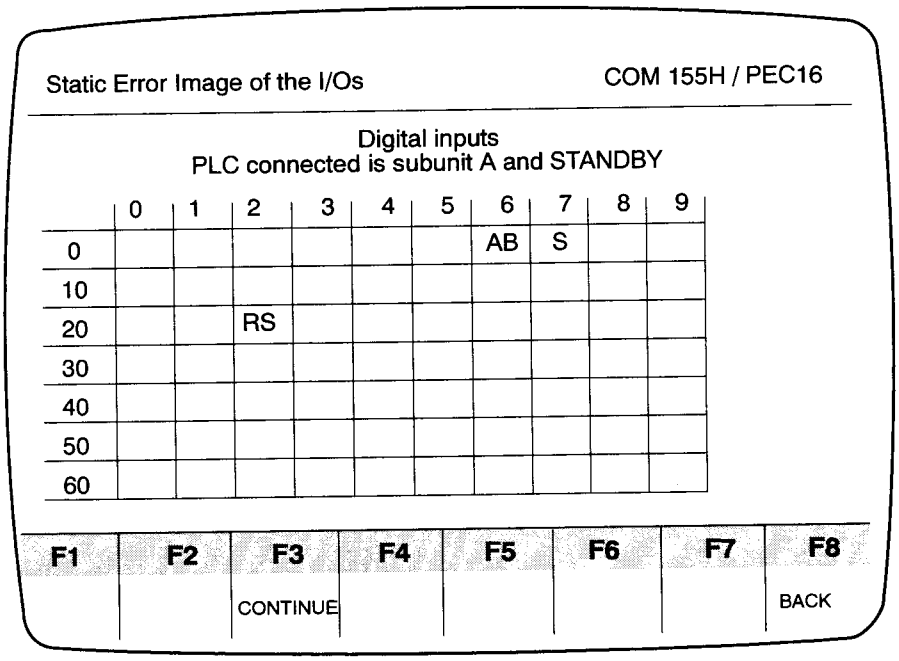


Figure 5-2 DI Error Image Basic Screen Form

F3 <CONTINUE> can be used, until DI 511 is reached. The following entries are possible:

- A Defect in subunit A
- B Defect in subunit B
- S Switched defective
- RA Configured redundant, defect in subunit A
- RB Configured redundant, defect in subunit B
- AB Configured redundant, A+B defective
- 3A 3-channel configured, defect in subunit A
- 3B 3-channel configured, defect in subunit B
- 3AB 3-channel configured, A+B defective
- R-A Readback DI/AI in subunit A defective
- R-B Readback DI/AI in subunit B defective
- RS Readback DI/AI in switched I/O defective
- 3-A 3rd channel in subunit A defective
- 3-B 3rd channel in subunit B defective
- 3-S 3rd channel in switched I/O defective
- L-A Locating DI in subunit A defective
- L-B Locating DI in subunit B defective
- L-AB Locating DI in subunit A and B defective

In the above example

- Input byte IB 6 in subunit A+B is defective
- Input byte IB 7 is switched defective
- Input byte IB 22 readback DI in switched I/O is defective

5

You can obtain the static error image of the expansion units by pressing F7 <EU> in the Basic Error Image form.

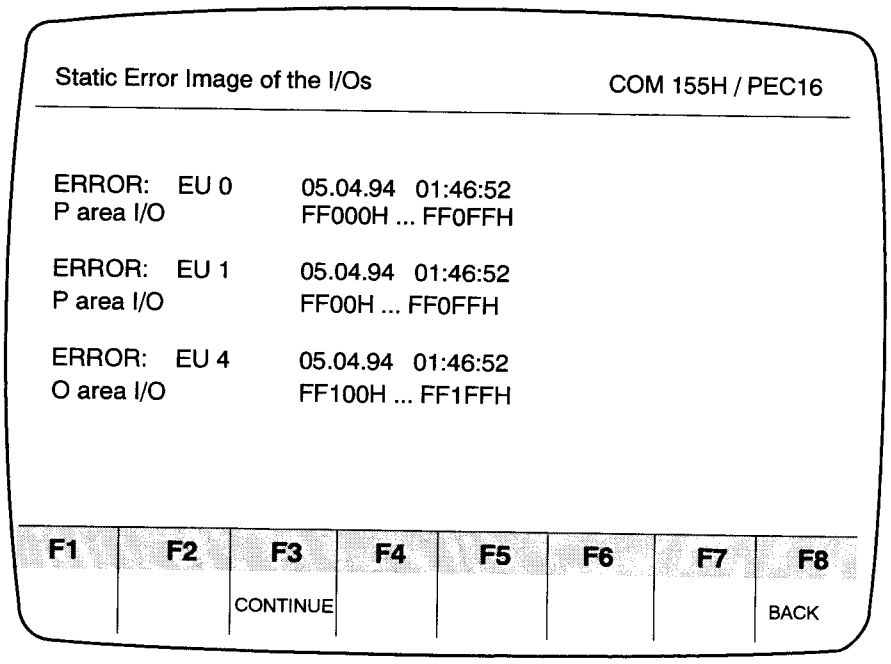


Figure 5-3 EU Error Image Basic Form

The numbers of the defective expansion units and their associated address spaces have been entered. The time of day only appears if the error is still entered in the error block. You can return to the basic form by pressing F8 <BACK>.

5.2 Error Data Block

Calling the Error DB

After pressing F4 <H ERROR> in the Diagnostics Menu, you must specify whether you wish to read the error data block from the PLC (with F1: on-line diagnostics) or from diskette (with F2: off-line diagnostics):

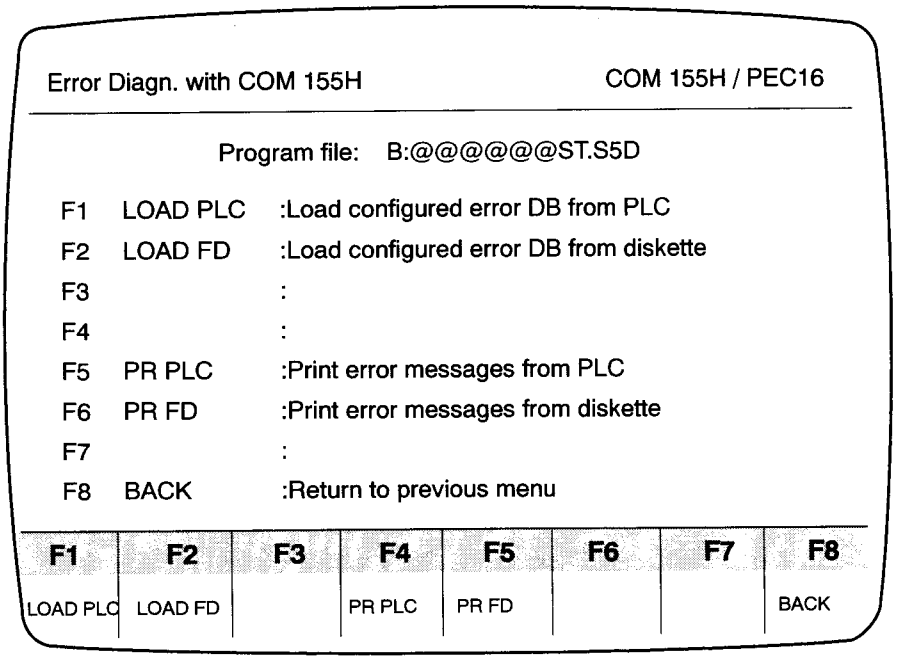


Figure 5-4 Error Diagnostics Screen Form

When you have selected the desired function, the contents of the error DB will appear on the screen. This applies to both subunits.

F1 <LOAD PLC>:
The error DB configured in DX1 is loaded.

If DX1 does not exist (after an overall reset, for example), the standard default error DB 3 is loaded. If there is no error DB 3, the programmer prompts you for a DB number. If this DB is also not in the programmable controller, the error DB is loaded from the absolute programmable controller address.

F2 <LOAD FD>:
The error DB configured in DX1 is loaded from the hard disk or from the diskette drive in the programmer.

F4 <PR PLC>:
All errors signalled in the programmable controller are printed in compressed form.

F5 <PR FD>:
All reported errors which are in the error DB of the file selected are printed in compressed form. The respective DX1 must also be available in the file.

Example of an Error Record Printout (Programmable Controller)

```
Error Diagnostics with COM 155H
COM 155H/PEC16
  SUBUNIT B

  ERROR RECORD NO.:2          CURRENT ERROR RECORD NO.: 3

  Error class     : Message
  Error : 95      : DI MODULE NOT CONFIGURED
  Time stamp      : 25.04.94  17:16:28
  DI ADDRESS      : F00B   DI BYTE NO. :11
  SUBUNIT B

  ERROR RECORD NO.: 3         CURRENT ERROR RECORD NO.: 3

  Error class     : Passivation
  Error:51        : TIMEOUT ON OUTPUT MODULES
  Time stamp      : 25.04.94  17:16:32
  DI ADDRESS      : F007   INSTR. CODE: 0000
```

Structure of the Error Diagnostics Screen Form

If several errors occur, the error last entered is displayed first. Each screen form corresponds to an error record in the error DB.

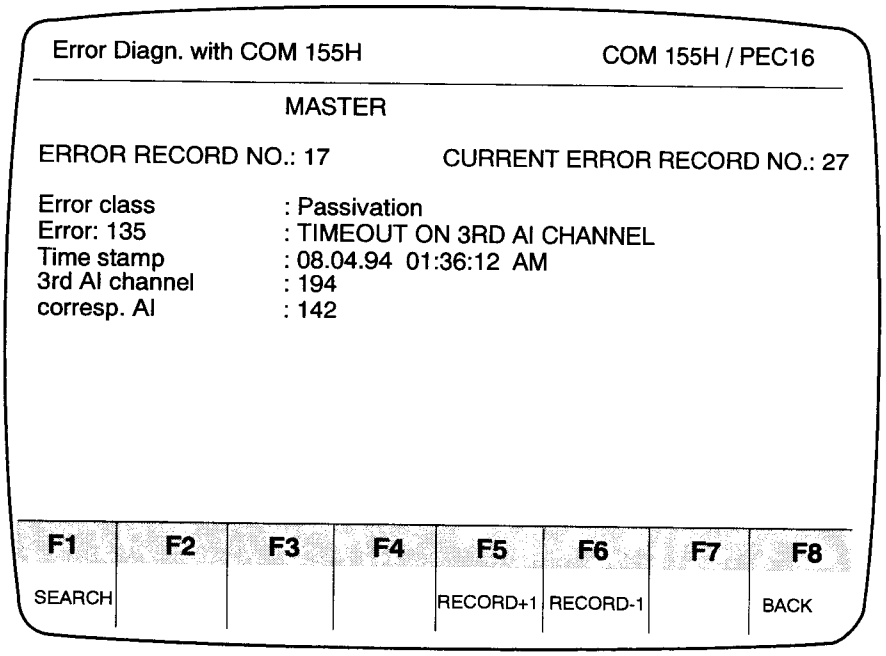


Figure 5-5 Error Diagnostics Screen Form (Example)

F1 <SEARCH>:

- If you wish to read out a particular error record quickly, press this key and enter the number of the desired record.
- If you want to find a particular error record quickly, press the <Cursor down> key and type in the error number.

F5 <RECORD+1> (or Cursor down ↓):
F6 <RECORD-1> (or Cursor up ↑):

These functions enable you to page through the error DB (forward and backward) record by record; all errors of both subunits stored up until that particular instant can be read out.

When the last error entered is read out, the message

"NO MORE ENTRIES"

appears.

Legend: Error DB

Each screen form tells you whether the error occurred in the MASTER or in the STANDBY.

ERROR RECORD NO. x:
Number of the error record you are presently reading out.

CURRENT ERROR RECORD NO. x:
The last stored error is located in error record number x.

Error class:
This indicates the standard error response (for example, hard STOP on a CPU fault, passivation in the case of a timeout, etc.).

Error:
The error number entered in the error DB is displayed in plaintext here (for example, I/O bus fault, timeout on output module, etc.).

Time stamp
Provided the CPU clock is set, the current date and time of day (from the system data area in the master CPU) appears here when a fault or error occurs.

All other specifications are supplementary information dependent on the errors or faults that have occurred (for example, instruction code, step address counter, EU number, etc.).

Please also refer to the structure of the error data block in Section 8.2 of the S5-155H Instructions.

5.3 Documenting with COM 155H

Printing the Configuration Overview

Press <F7> in the Main Menu to get into the "System Handling" menu. The <F1> key takes you into the Directory menu. From this menu, you can print out an overview of your configuration in the form of a list. With <F4> you get the configuration from the programmable controller, with <F5> from the diskette.

Example:

I/O group	! I/O type	! Subunit	! Number
Digital inputs	! one-sided I/O	! A	! 2
Digital inputs	! one-sided I/O	! B	! 2
Digital inputs	! switched I/O	!	! 36
Digital inputs	! redundant I/O	!	! 6
Digital inputs	! 3-channel I/O	!	! 3
Digital outputs	! one-sided I/O	! A	! 4
Digital outputs	! one-sided I/O	! B	! 0
Digital outputs	! switched I/O	!	! 40
Digital outputs	! redundant I/O	!	! 13
Analog inputs	! one-sided I/O	! A	! 2
Analog inputs	! one-sided I/O	! B	! 0
Analog inputs	! switched I/O	!	! 14
Analog inputs	! redundant I/O	!	! 8
Analog inputs	! 3-channel I/O	!	! 6
Analog outputs	! one-sided I/O	! A	! 0
Analog outputs	! one-sided I/O	! B	! 8
Analog outputs	! switched I/O	!	! 8
Analog outputs	! redundant I/O	!	! 8
CP/IP interfaces	! one-sided I/O	! A	! 1
CP/IP interfaces	! one-sided I/O	! B	! 1
CP/IP interfaces	! switched I/O	!	! 9

Printing the Configuration DB/DX

Press <F4> in the "System Handling" menu to get into the COM 155H Print Menu. You can have your configuration printed out in tabular form from this menu.

If the data is to be printed out from the programmable controller, diskette or EPROM flash memory card, it must first be loaded into the programmer's memory. This is done from the System Handling menu, F3 <TRAN LOAD>.

A footer is printed out on each page.

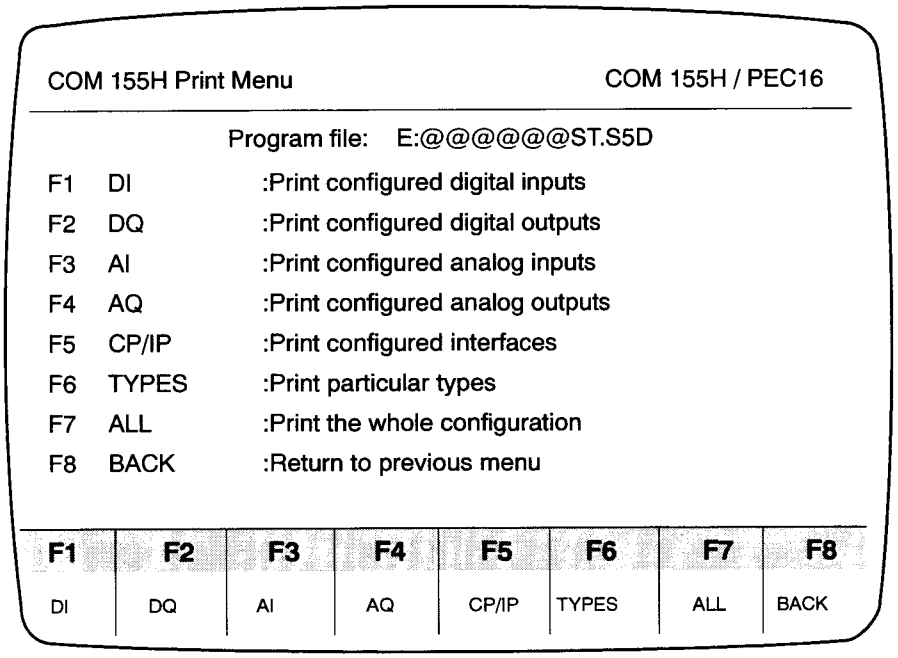


Figure 5-6 Print Menu

Functions in the Print Menu

You can execute the following functions using <F1> to <F7>:

F1 <DI>: Print out all DI types (digital inputs)

F2 <DQ>: Print out all DQ types (digital outputs)

F3 <AI>: Print out all AI types (analog inputs)

F4 <AQ>: Print out all AQ types (analog outputs)

Example: Table of DI Types (F1)

I/O byte	Short symbol	Type	Subunit	I/O byte	Short symbol	Type	Subunit
IB 0		2		IB 1		2	
IB 126		L-DI		IB 127		L-DI	

5

F5 <CP/IP>: Print all CP and IP types (interface no.)

Example: Table of CP/IP Types (F5)

Interface number	Type	Subunit	Interface number	Type	Subunit
0	24	A	10	24	B

F6:
<SELECT> Printout of individual types, printout of the configured L-DIs, L-DQs and R-DIs (bit assignment) or printout of your operating system parameter settings.

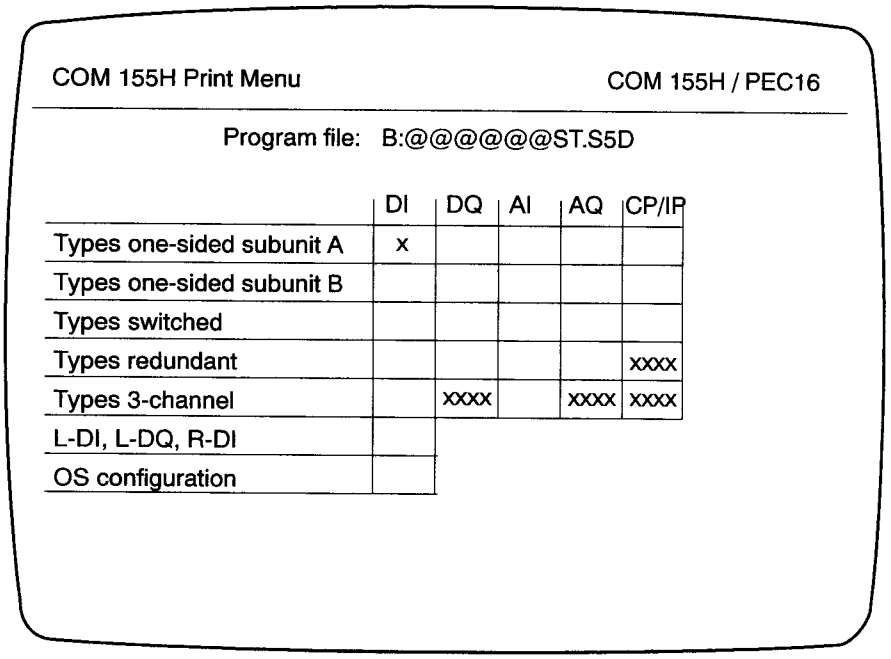


Figure 5-7 Print Menu (TYPES, F6)

- Position the cursor to the desired field and press <RETURN> or <ESC>. What you have selected will be output to the printer.

Example: Types; redundant DI

DI type 3: Two-channel digital inputs "redundant I/O"

DI byte	Short symbol	Discrep. time (sec)	L-DQ byte/bit	L-DI byte/bit
I 5.0		0.05	130.5	6.2
I 5.1		0.05	130.5	6.2
I 5.2		0.05	130.5	6.2

Example: L-DI, L-DQ, R-DI

Bit assignment digital outputs:

Bits ->	! 0	! 1	! 2	! 3	! 3	! 5	! 6	! 7
DQ byte 130	!	!	!	!	!	!DI-LDQ	!	!

Bit 5 of digital output byte 130 has been configured as a locating digital output for a redundant digital input. The other bits are still unassigned.

If you mark the OS configuration field in the Print Menu with a cross ("check" it), you get a printout of the following:

1. The operating system parameters
2. The transfer data for standby activation and
3. The I/O areas of the expansion units.

F7 <ALL>: The entire configuration is printed out.

This function enables you to print out out all the data of your configuration:

1. The operating system parameters (see above) and
2. The entire I/O configuration.

Abbreviations

A

This appendix lists and explains the abbreviations and mnemonics used throughout this manual.

A-NR	Job number
ABS	Absolute value
ADF	Addressing error
AI	Analog input
AGF	PLC error
AMA	Subunit is master
ANZW	Condition codeword
AQ	Analog output
AS	Absolute addressing
BASP	Command output disable
BCD	Binary coded decimal number
BLGR	Frame size
BS	System data area
BSTACK	Block stack
BSU	Bit slice unit
C	Counter
CC	Central controller
CP	Communications processor
CPU	Central processing unit
DB	Data block
DBNR	Data block no., DX no., no. of the address area (for IA operation)
DHB	Standard data handling block
DI	Digital input
DQ	Digital output
DW	Data word
DX	Extended data block
E-DB	Error data block
EPROM	Erasable programmable read-only memory
ESC	Escape
EU	Expansion unit
F	Flag (bit)
FB	Function block
FD	Floppy disk (diskette, drive)
FX	Extended function block

G DB	STEP 5 operation "Generate data block"
GP	Global I/Os, virtual memory area that can be mapped to unassigned I/O areas of the individual SINEC L2 stations
HW	Hardware
I	Input module
I/O	Input/output
IB	Input byte, parameter for source/destination data of the process input image
IF	Interface
IM	Interface module
INT	Interrupt
IP	Intelligent input/output module
ISTACK	Interrupt stack
KF	Fixed-point constant
L-DI	Locating digital input
L-DQ	Locating digital output
LAN	Local area network
LED	Light-emitting diode
MAI	Memory and interface
MB	Flag area (F)
MFDT	Mean failure detection time
MTTF	Mean time to failure
MUART	Multifunction universal asynchronous receiver transmitter
NAU	Power failure
NN	No source/destination parameters at block
OB	Organization block
O-I/O	Extended I/O area (F F100 ... F F1FF)
OS	Operating system
OY	Peripheral byte (extended)
P-I/O	I/O area (F F000 ... F F0FF)
PAFE	Parameter assignment error
PARE	Parity error
PB	Parameter for source/destination data from/in I/O modules
PEU	EU power supply failure
PG	Programmer

PI	Process image
PII	Process input image
PIQ	Process output image
PLC	Programmable controller
PY	Periferal byte
Q	Output module
QANF	Relative start address within the type
QB	Output byte, parameter for source/target data of the process output image
QLAE	Number (length) of source data
QTYP	Type of data source
QVZ	Timeout
R-DI	Readback digital input
RALU	Registered Arithmetic Logic Unit
RAM	Random access memory
REL	Relative value in %
RLO	Result of logic operation
RT	Extended system data area
RW	Read/write
SAC	STEP 5 address counter
SB	Sequence block
SPS	Standard slot
SS	Interface
SSNR	Interface number
STP	Stop at end of cycle
STS	Direct stop
STUEB	Block stack overflow
STUEU	Interrupt stack overflow
SUF	Substitution error
T	Timer
TLAF	Transfer load error
XX	Type identifier for indirect parameter assignment
ZANF	Relative start address within the type
ZLAE	Number (length) of target data
ZTYP	Type of data target

Index

B

Note:
The page numbers following the terms and search keys of this index are preceded by a I, II or III (Roman numerals), which stands for the part of the manual containing the corresponding information.

The individual parts of the manual are separated by a red page.

A

Absolute addresses, II/3-6

Address area, II/3-6

AI, redundant, I/6-16

AI, redundant, I/4-8
- calculating XA, I/4-22
- channel types, I/4-22
- discrepancy, I/4-24
- error detection, I/4-24
- example, discrepancy calculation, I/4-24
- FB 40, I/4-20
- range limits of XA, I/4-22
- range violation BU, I/4-22
- selective sampling, I/4-23
- XA in the event of a wirebreak, I/4-22

AI, redundant, three-channel
- calculating XA, I/4-27
- channel type, I/4-26
- error detection, I/4-28
- example, discrepancy calculation, I/4-28
- FB 43, I/4-25
- range limits of XA, I/4-27
- range violation BU, I/4-27
- selective sampling, I/4-27
- XA in the event of a wirebreak, I/4-27

AI, three-channel
- with one sensor, I/4-29
- with three sensors, I/4-29

Analog inputs, III/4-11
- legend, III/4-13

Analog outputs, III/4-14

Analog value, I/4-22, I/4-26

ANZW, II/2-16

Application,
- SEND-block, II/4-4

Application examples, I/10-1
- hardware, I/10-2
- I/Os, one-sided, I/10-11
- I/Os, redundant, I/10-12
- I/Os, switched, I/10-6
- jumper configuration on IM 304, I/10-4
- point-to-point link, I/10-17
- software, I/10-2
- tasks, I/10-2

Applications, I/1-3

AQ, redundant, I/4-30, I/6-16

AQ, redundant, I/4-8
- direct I/O access, I/4-31
- fault tolerance, I/4-30
- two-channel, I/4-30

Area length remaining, II/3-5

Area limits, II/3-5

B

Block
- CONTROL, II/4-12
- FETCH, II/4-10
- REC-A, II/4-9
- RECEIVE, II/4-6
- RESET, II/4-13
- SEND, II/4-2
- SEND-A, II/4-5
- SYNCHRON, II/4-14

Block idling, II/1-5, II/4-11

Block number, I/6-12, I/6-13

C

Cable 721, I/6-13

Calling data handling blocks, II/3-2

Central processing units (CPU), replacing of, I/9-2

Channel type, I/4-22, I/4-26

Characteristics field, III/4-4

Circuits, I/6-3

Cold restart, II/1-5

Cold restart with memory, II/1-5

COM 155H, I/6-8
- calling, I/10-6
- configuration, III/1-2
- configuring (one-sided) DQs, I/10-11
- configuring (switched) DIs, I/10-9
- configuring (switched) DQs, I/10-9
- configuring and error diagnostics, I/1-10
- defaults screen form, III/1-4
- initializing operating system, I/10-7
- operator prompting, III/1-2
- scope of supply, III/1-2
- setting system size, I/10-8
- start, III/1-4, III/1-8
- transfer DX 1, I/10-10

Communication, II/3-3

Communications processors (CPs), I/5-3

Condition codeword, II/2-19, II/3-4
- division, II/2-25
- operation, II/2-25
- structure, II/2-23

Condition codeword: ANZW, II/2-7
Configuration, I/6-2
  hybrid, redundant, I/1-6
  one-channel, I/1-4
  three-channel, redundant, I/1-5
  two–channel, redundant, I/1-5
Configuration DB, printing, III/5-9
Configuration screen form, III/3-2
Configuration versions
  hybrid I/Os, I/4-42
  one-sided I/Os, I/4-38
  redundant I/Os, I/4-6
  switched I/Os, I/4-30
Configuring, I/1-10
  data source, III/2-2, III/2-3
  printing the configuration overview, III/5-8
Configuring block DX 1, I/12-2
Configuring data,
  printing, I/6-16
Connecting cable,
  replacing of, I/9-3
  type 721, I/1-11, I/6-3
CONTROL, II/4-12
Control byte, I/2-5, I/12-3
Control elements, I/3-4
Control form, III/2-8
CONTROL-ALL, II/4-12
CONTROL-DIRECT, II/4-12
Counter/timer cells ZB/TB, II/3-5
CP
  depassivation of interfaces, II/1-5
  passivation, II/1-5
  redundant, I/5-5
CP 143, II/4-4
CP module, II/1-2
CP/IP interface, III/4-18
CPU 948 R,
  configuration, I/3-2
CPU 948R, I/3-1
  application, I/3-2
  cold restart, I/3-5, I/3-8
  cold restart with memory function, I/3-5, I/3-8
  error LEDs, I/3-7
  interface fault LEDs, I/3-7
  overall reset, I/3-5, I/3-8
  removing and inserting the module, I/3-3
  startup, I/3-7
CPU real-time clock, III/3-7
Cycle errors, I/2-9
Cycle time, I/1-10, III/3-6
Cycle time extension, I/7-4

D

Data block
  in the case of indirect parameterization, II/2-15
  in the case of READ/WRITE, II/2-15
Data blocks, II/3-4
Data exchange, II/1-5
Data handling block
  calling, II/3-2
  runtime, II/3-8
Data handling blocks
  application, II/1-2
  calling, II/1-3
  condition code evaluation, II/1-3
  existing function blocks, II/1-4
  functions, II/1-4, II/2-2
  parameters, II/2-2
  special features, II/1-3
Data interchange, I/10-21, II/1-2
Data source/data destination, II/2-12
DB/DX data blocks, III/3-9
DB/DX number, III/3-10
Defaults, III/2-9
Delete menu
  DX 1, III/2-11
  types, III/2-11
Depassivation, I/12-2
Destination parameters, example, II/2-11
DHB call-up
  conditions, II/3-2
  in program, II/3-2
DHB SYNCHRON
  in cycle, II/4-15
  in restart, II/4-14
DI 0
  configuring, I/4-14
  wiring, I/4-14
DI, redundant
  redundant, I/6-15
  three-channel, I/4-12
  with error locating facility, I/2-13, I/4-7, I/4-10
  without error locating, I/4-7, I/4-10
DI, switched,
  configuration, I/10-9

Diagnostics,
basic menu, III/2-6
Diagnostics functions, III/2-6
Digital and analog I/Os, I/4-6
Digital inputs (DI), III/4-6
legend, III/4-8
type 1, III/4-6
type 2, III/4-7
type 3, III/4-7
type 4, III/4-8
Digital outputs (DQ), III/4-9
legend, III/4-11
type 10, III/4-10
type 11, III/4-10
type 8, III/4-9
type 9, III/4-9
Direct I/O access, I/4-10, I/4-11, I/4-13, I/4-20
Direct parameterization, II/2-3
Directory menu, III/2-10
Discrepancy error, I/4-24, I/4-28
Discrepancy monitoring, I/1-9
Discrepancy time, I/4-10, I/4-14, I/12-2, III/3-7, III/4-8
Discrepancy value, I/4-28, I/12-2, III/4-13
Documenting, III/5-8
DQ,
155H system program response, I/4-18
DQ,
error detection, I/4-9
redundant, I/6-16
with error locating facility, I/2-13, I/4-16, I/4-18
with intermittent outputs, I/4-16
with non-intermittent outputs, I/4-18
without locating facility, I/4-14
without/with error locating, I/4-8
DQ, switched,
configuration, I/10-9
Dual-port RAM, II/1-2
addressing, II/1-4
Dynamic response,
during standby activation, I/7-4

E

Equipotential bonding conductor, I/6-3
Error, III/5-7
Error (line), III/4-4
Error class, I/8-8, III/5-7
Error counter, I/8-6
Error data block, III/5-5
Error data block (E-DB), I/8-4
entries, I/8-4
error counter, I/8-6
evaluation, I/8-24
status word, I/8-6
structure, I/8-5
Error DB
basic diagnostics menu, III/5-5
calling, III/5-5
legend, III/5-7
Error DB number, III/3-6
Error detection time, III/3-6
Error diagnostics, I/1-10, I/8-23
Error diagnostics screen form, III/5-6
Error handling, I/8-3
Error image, I/8-4, I/8-7, III/5-2
STAT ERR, III/2-6
Error locating, I/2-11, I/2-13, I/12-2
Error locating facility, I/4-7
for intermittent DQs, I/4-19
for redundant DIs, I/4-11
for redundant DQs, I/4-17, I/4-31, I/4-33
Error location, I/8-8
additional information, I/8-9
Error message list, I/8-11
error no. ... 139, I/8-21
error no. 1 ..., I/8-11
Error messages
additional information, I/8-22, I/8-23
output via FB 48, I/8-25
Error number, I/8-9
Error OB 37, I/8-3, I/8-24, I/8-28
operating system error messages, I/8-24
Error record, I/8-4
in diagnostics screen form, III/5-6
number, III/5-7
printing, III/5-6
search, III/5-6
structure, I/8-8
Error recovery (error messages), I/8-2
Error search mode, I/1-8, I/2-13, I/12-2
calling, I/2-13
sequence, I/2-13

Error word (in FB 48), I/8-26
EU
  IM and EU switched, I/4-41
  one-sided, I/4-38
  redundant, I/4-6
  switched, I/4-40
EU number, I/6-12

F

Failsafety, I/1-2
Failure and repair
  of CP/IP modules, I/9-6
  of EU, I/9-4
  of I/O modules, I/9-5
Failure subunit, I/Os, I/9-9
Fault tolerance, I/2-11
Fault-tolerance systems, I/1-3
FB 40
  calling, I/4-20
  input/output parameters, I/4-21
FB 41, I/4-34
  calling, I/4-36
  input/output parameters, I/4-37
FB 43
  calling, I/4-25
  input/output parameters, I/4-26
FB 48, I/8-25
FETCH, II/4-10
Fetch job, II/4-10
FETCH-DIRECT, II/4-10
Firmware version, III/1-14
Flag doubleword, III/3-7
Frame size, II/4-17
Frame size: BLGR, II/2-8
Frontplate, I/3-4
Function
  SEND-A, II/4-5
  SEND-ALL, II/4-3
  SEND-DIRECT, II/4-3

G

Group power supply, I/4-16, I/12-2

H

H error DB, III/3-7
H flag control byte, II/1-5
H flag doubleword, I/8-3, I/8-27, I/12-3, III/3-7
  application, I/8-27
H flag word, I/4-14, I/8-3, I/8-29, I/12-3
  control byte, I/8-30
  status byte, I/8-29
  structure, I/8-29
H system error, III/3-6
H system flag word, III/3-7
Handshake, II/1-5, II/3-8, II/4-3, II/4-4, II/4-8, II/4-11
Hybrid I/O configuration, I/4-42

I

I/O
  copy, III/4-4
  delete, III/4-5
  search, III/4-4
  select, III/4-5
  swap, III/4-5
I/O address areas, I/4-4, I/6-13
  redundant I/Os, I/4-7
I/O areas of the EUs (IM 314), III/3-11
I/O bus cable 721, I/2-12
I/O configuration screen form, structure, III/4-2
I/O configuration types, III/4-6
I/O direct access, I/4-10, I/4-11, I/4-13, I/4-20
I/O error image, III/5-2
I/O modules, one-sided, I/4-38
I/O modules, switched, I/4-40
I/O operating modes, I/4-2
  hybrid, I/4-2
  one-sided, I/5-3
  switched, I/5-4
I/O types, I/4-2, I/4-3, III/4-3
  printing, III/5-9
I/Os, I/4-7
  address areas, I/4-7
  basic screen form, III/3-4
  configuring, I/4-3
  one-sided, I/4-2, I/4-38

one-sided, I/O modules, I/4-38
one-sided, IM and EU, I/4-38
one-sided, program example, I/4-39
redundant, I/4-2, I/4-6, I/12-4
redundant, three-channel, I/4-2
switched, I/4-2, I/4-40
switched configuration, I/5-4
switched, I/O modules, I/4-40

I/Os, configuring
one-sided, I/6-8
redundant, I/6-15
switched, I/6-10

I/Os, one-sided
configuring, I/10-11
configuring DQs, I/10-11
initializing, I/6-8

I/Os, redundant
configuring, I/6-15, I/10-12
configuring DIs, I/10-15
configuring DQs, I/10-15
interconnecting, I/10-12
printing out the configuration data, I/10-16
putting into operation, I/6-16
setting the system size, I/10-14

I/Os, switched
configuring, I/10-6
initialization, I/6-13
on-line functions, I/10-10
PLC startup, I/10-10
putting into operation, I/6-14
setting the EU number, I/6-12
statement execution time for access operations, I/7-2

IM 304, jumper configuration, I/6-5, I/6-10

IM 304 – IM 314R, I/6-10

IM 304 – IM 314R, I/4-40

IM 314R, I/6-11, III/3-11
command output inhibit (BASP), I/6-13
connector pin assignments, I/11-3
jumper configuration, I/6-11
LEDs on the frontplate, I/6-12
technical specifications, I/11-2

IM 324R
coding plug, I/11-7
connector pin assignment, I/11-6
jumper configuration, I/6-6
technical specifications, I/11-5

Indirect parameterization, II/2-3
DB structure, II/2-5

Interface,
initializing, II/1-5

Interface module
IM 304, I/6-5
IM 314R, I/6-11
IM 324R, I/6-6

Interface modules
one-sided, I/4-38
redundant, I/4-6
replacing, I/9-2
switched, I/4-40

Interface number: SSNR, II/2-7

Intermittent outputs, I/12-3

Interrupt DI, III/3-8

Interrupt processing, response time, I/7-5

IPs,
switched, I/5-4

J

Job number, II/2-20

Job number: A-NR, II/2-7

Job status, II/2-22, II/4-12

Jumper configuration
IM 304, I/6-5, I/6-10
IM 314R, I/6-11
IM 324R, I/6-6

K

Key (in basic screen form)
I/O 314, III/3-11
SYSTEM, III/3-5
TRAFDAT, III/3-9

L

L-DI, I/10-15

L-DQ, I/4-11, I/10-15

LED
during restart, I/2-11
during self-test, I/2-11
during startup, I/6-9, I/6-14, I/6-16
with standby activated, I/2-5

Legend
analog inputs, III/4-13
digital inputs, III/4-8
digital outputs, III/4-11

Load/transfer menu, III/2-12
Locating DI/DQ, III/4-8, III/4-11
Locating digital input (L-DI), I/2-13, I/4-7
Locating digital output (L-DQ), I/2-13, I/4-7
Locating facility (LF), I/2-13, I/4-7

M

Main menu screen form, III/2-2
Master-standby switchover, I/2-8
Maximum configuration, I/1-11
Minimum configuration, I/1-11, I/6-7
Mode selector switch, I/3-5

N

NON-STOP DI, I/4-7
NON-STOP DQ, I/4-7
NON-STOP operation, I/1-2, I/4-6
NON-STOP-operation, I/12-3

O

On-line functions, 'Start', I/2-4
One-sided I/Os
  initializing, I/6-8
  putting into operation, I/6-9
Operating states, I/1-8, I/2-2
Operating system
  basic screen form, III/3-3, III/3-5
  explanation of parameters, III/3-6
  initializing, I/10-7, III/3-5
  screen form, III/3-5
Operation
  mode selector switch, I/3-5
  reset switch, I/3-5
  status indicators (LEDs), I/3-5
Output parameters,
  evaluation, II/2-16
Output value XA, I/4-21, I/4-26
Outputs,
  intermittent, I/4-16
Overflow,
  overrange BU, I/4-27, I/4-28

P

PAFE, II/2-16
PAFE Byte, II/2-9
PAFE byte, II/2-17
Page addressing, I/2-12
Parallel link CC – CC, I/10-3
Parallel link IM 304 – IM 324R,
  configuring, I/6-5
Parallel link IM 314 – IM 324R, I/1-11
Parallel link IM304 – IM324R, I/2-12
Parameter description, II/2-7
Parameter: ANZW, II/2-19
Parameterization
  direct/indirect, II/2-3
  examples, II/2-4
Parameterization error: PAFE, II/2-8
Parameterization types, II/2-3
Parameters
  "frame size", II/4-17
  A-NR, II/2-7
  ANZW, II/2-7
  BLGR, II/2-8
  SSNR, II/2-7
Parameters of
  FB CONTROL (FB123), II/4-12
  FB FETCH (FB122), II/4-10
  FB REC-A (FB127), II/4-9
  FB RECEIVE (FB121), II/4-6
  FB RESET (FB124), II/4-13
  FB SEND (FB120), II/4-2
  FB SEND-A (FB126), II/4-5
  FB SYNCHRON (FB125), II/4-14
Passivation, I/4-24, I/4-28, I/12-3
PLC
  INFO, III/2-8
  start, III/2-4
  stop, III/2-4
PLC functions, III/2-4
Point-to-point link
  data interchange, I/10-19
  data test, I/10-21
  initializing, I/10-18
  reactivationg a failed CP, I/10-26
  restart routine, I/10-18
  switched, data interchange, I/10-25

switched, parameter initialization, I/10-25
switched, PLC restart, I/10-25
switched, redundant, I/10-23
two-channel, redundant, I/10-17
Power supply block 955, I/6-4
Power supply units, I/6-3
Print menu, III/2-15, III/5-9
types, III/5-10
Process image
of the inputs (PII), I/1-9
of the outputs (PIQ), I/1-10, I/2-13
updating, I/2-9
updating of the PII/PIQ, I/4-5
Process interrupt, I/2-6, I/2-12
Process interrupt, redundant
DI 0, I/4-14
function and error handling, I/4-14
Program execution levels, II/3-2
Program sequence, II/1-5
Programming, I/1-9

Q

Quantity of transmitted data, II/2-21

R

RAM data block, III/3-6
Range violation, I/4-22, I/4-27
Readback delay, I/12-4, III/3-8
Readback delay time, I/4-16, I/4-18, I/11-8
Readback DI, III/4-11
Readback digital inputs (R-DI), I/4-14, I/12-4
REC-A, II/4-9
RECEIVE, II/4-6
RECEIVE-ALL, II/4-7
RECEIVE-DIRECT, II/4-8
Redundant I/Os, I/12-4
Redundant mode, I/1-8, I/12-4
with CP, I/5-5
Repair
of connecting cable 721, I/9-3
of CP/IP modules, I/9-6
of EU, I/9-4
of I/O modules, I/9-5
of the CPU 948R, I/9-2
of the parallel link IM 304/IM 324R, I/9-2
Replacement
721 connecting cable, I/9-3
central processing unit (CPU), I/9-2
CP/IP modules, I/9-6
I/O modules, I/9-5
interface modules (IM), I/9-2
RESET, II/4-13
Reset switch, I/3-5
RESET-ALL, II/4-13
RESET-DIRECT, II/4-13
Response to faults, I/9-1
Restart self-test, I/2-5
Restart time, I/7-3
Ring selection, III/4-5
RLO, II/2-16
Runtime, II/3-8
Runtimes, I/7-3
155H system program, I/7-3
I/O test, I/7-3
self-test, I/7-3

S

S5 communications, II/1-2
S5-155H programmable controller
applications, I/1-3
characteristics and functions, I/1-2
hardware configuration, I/1-11
infeed to power supply units, I/6-3
installation recommendations and rules, I/6-3
method of operation, I/1-7, I/1-8, I/2-2
operating states, I/1-8, I/2-2
performance characteristics, I/1-2
program processing, I/1-9
response to faults/errors, I/6-17
specific functions, I/1-7
structure, I/1-4, I/6-2
system performance characteristics, I/1-2
Sampling
cyclic, I/4-20
selective, I/4-20, I/4-23
Screen form
configuring, III/3-2
configuring the I/O, III/3-4
control, III/2-8
delete, III/2-11
diagnostics, III/2-6
directory, III/2-10
flag word, III/2-7

main menu, III/2-2
operating system, III/3-3
PLC functions, III/2-4
print menu, III/2-14, III/2-15
printing, III/5-9
status, III/2-7
system handling, III/2-9
transfer data, III/3-9

Screen form tree, III/1-6

Scroll function, III/4-4

Selective sampling
AI, I/4-21
AI, three-channel, I/4-26

Self-test, I/2-11, I/12-4
execution time, I/2-12
fault tolerance, I/2-11
in error search mode, I/2-13
in restart routine, I/2-11
strategy, I/2-11
system components, I/2-12

Self-test functions, I/1-9

Self-test routine, III/3-6

SEND, II/4-2

SEND-A, II/4-5

SEND-ALL, II/4-3

SEND-ALL/RECEIVE-ALL, calling, II/3-3

SEND-DIRECT, II/4-3

SINEC-CP, redundant, I/5-5

Software change, III/2-5

Solo mode, I/1-8, I/12-5

Source parameters, example, II/2-10

Source/destination data frame
length, II/2-12
start address, II/2-12
type, II/2-12

Source/destination parameters, II/2-10
meaning, II/2-13

Standard function blocks
FB 40, I/4-21
FB 41, I/4-34
FB 43, I/4-26

Standby activation, I/1-8, I/2-2, I/2-5, I/12-4, III/3-9
dynamic response, I/7-4
increase in cycle time, I/7-4
process, I/2-5

Standby-master switchover, I/4-14, I/4-40, I/5-4, I/9-8, I/12-4
bumpless, I/9-9
switching criteria, I/9-8
switchover time, I/9-9
when master CC fails, I/9-8

Standby, updating
disabling, I/2-5
enabling, I/2-5
procedure, I/2-7
transfer data, I/2-6

Starting COM 155H, III/1-4, III/1-8

Startup of the S5-155H programmable controller, I/6-9, I/6-14, I/6-16

Statement execution times, I/7-2

Status (line), III/4-4

Status byte, I/12-3

Status indicators, I/3-5

Status screen form, III/2-7

Status word, I/8-4, I/8-6

STEP 5 programm processing, I/2-8
restrictions, I/1-14

STEP 5 user program, I/1-13
special functions, I/1-14
startup, I/1-14

STEP address counter, I/8-22, I/8-23

Stuck at 0 error, I/4-9, I/4-14, I/4-16, I/12-5

Stuck at 1 error, I/4-9, I/4-14, I/4-18, I/12-5

Switched I/Os,, initializing, I/6-13

Switchover time, I/9-9

Symbols line, III/4-3

SYNCHRON, II/4-14

Synchronization, II/4-15
clock, I/2-9
interrupt servicing, I/2-8
program processing, I/2-8
system monitoring, I/2-8

Synchronization error, I/2-8

Synchronization point, I/2-8, I/12-5

Synchronization principle, I/2-8

System
fail-safe, I/1-3
fault–tolerant, I/1-3

System data area RS, II/3-5

System failure, I/1-2
System flag word, III/3-7
System handling, III/2-9
System handling menu, III/2-9
System program 155H, I/1-13
System program runtimes, I/7-3
System size,
  setting of, I/10-8

T

Technical specifications, I/11-1
Terminator, I/6-12, I/6-13
Test execution time, I/2-12
Test slices, I/12-5
  number of, III/3-6
Time, III/3-7
Time stamp, I/8-10, I/8-27, III/3-7, III/5-7
Timed interrupt, I/2-6, I/2-12
Transfer data, I/12-5, III/3-9
Transfer data screen form, III/3-9
Transfer menu, III/2-13
Troubleshooting, I/8-2, I/8-3
Type number, III/4-2

U

Updating, I/12-5
  dynamic data, I/2-7
  user program, I/2-7
Updating procedure, I/2-5
User program, processing, I/2-3

W

Wirebreak, I/4-28
Write pointer, I/8-6

Siemens AG
AUT 1282
Östliche Rheinbrückenstr. 50
D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: .....
Your Title: .....
Company Name: .....
Street: .....
City, Zip Code: .....
Phone: .....

Please check any industry that applies to you:

- ❑ Automotive
- ❑ Chemical
- ❑ Electrical Machinery
- ❑ Food
- ❑ Instrument and Control
- ❑ Nonelectrical Machinery
- ❑ Petrochemical
- ❑ Pharmaceutical
- ❑ Plastic
- ❑ Pulp and Paper
- ❑ Textiles
- ❑ Transportation
- ❑ Other



Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1.Do the contents meet your requirements? ☐

2.Is the information you need easy to find? ☐

3.Is the text easy to understand? ☐

4.Does the level of technical detail meet your requirements? ☐

5.Please rate the quality of the graphics/tables: ☐

Additional comments:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Siemens AG
AUT 1282
Östliche Rheinbrückenstr. 50
D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: .....
Your Title: .....
Company Name: .....
Street: .....
City, Zip Code: .....
Phone: .....

Please check any industry that applies to you:

- ❑ Automotive
- ❑ Chemical
- ❑ Electrical Machinery
- ❑ Food
- ❑ Instrument and Control
- ❑ Nonelectrical Machinery
- ❑ Petrochemical
- ❑ Pharmaceutical
- ❑ Plastic
- ❑ Pulp and Paper
- ❑ Textiles
- ❑ Transportation
- ❑ Other



Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1.Do the contents meet your requirements? ☐

2.Is the information you need easy to find? ☐

3.Is the text easy to understand? ☐

4.Does the level of technical detail meet your requirements? ☐

5.Please rate the quality of the graphics/tables: ☐

Additional comments:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

SIEMENS

SIMATIC S5

S5-155H Programmable Controller (CPU 948R)

Programming Guide

Volume 2/2

Order No. 6ES5 998-4SR21
Release 02

Introduction 1

User Program 2

Program Execution 3

Operating Statuses and Program Execution Levels 4

Interrupt and Error Diagnosis 5

Integrated Special Functions 6

Extended Data Block DX 0 7

Memory Assignment and Memory Organization 8

Memory Access Using Absolute Addresses 9

PG Interfaces and Functions 10

Appendix 11

Further Reading 12

List of Abbreviations, List of Keywords, List of Tables and Figures 13

The Pocket Guide CPU 948R

Order No. 6ES5 997-3UR21

is included in this manual.

C79000-H8576-C136-02

Copyright

Copyright © Siemens AG 1994 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.

Technical data subject to change.

Safety-related 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:



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.

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 equipment, systems and circuits in accordance with established safety practices and standards.

Siemens Aktiengesellschaft

6ES5 998-4SR21
EWK Elektronikwerk Karlsruhe

Printed in the Federal Republic of Germany

How to Use this Manual

Scope

This programming guide describes the following models of the CPU 948R and its system software:

Models of the CPU 948R

- CPU 948R-1 with 640 Kbytes of user memory,
  Order no. 6ES5 948-**3UR11,** version 01 and higher

- CPU 948R-2 with 1 664 Kbytes of user memory,
  Order no. 6ES5 948-**3UR21**, version 01 and higher

Overview of the Chapters

Chapter 1

This explains the typical mode of operation of the CPU and illustrates how a CPU program is structured.
The chapter also contains suggestions about how to tackle programming and which characteristics of the CPU 948R are important for programming.
If you have already worked with the CPU 946R/947R and want to know the differences between these CPUs and the CPU 948R you will find this information in this chapter.

Chapter 2

This explains the components of a STEP 5 user program and how the program can be structured.

Chapter 3

This is intended for readers who do not yet have much experience of using the STEP 5 programming language. It therefore deals with the basics of STEP 5 programming and explains the STEP 5 operations in detail (with examples).

Experienced readers who may find that the information about specific operations in the pocket guide is inadequate, can use Section 3.5 as a reference section.

Chapter 4

This provides an overview of the modes and program execution levels of the CPU 948R. It provides you with detailed information about various start-up modes and the associated organization blocks in which you can program your routines for differrent start-up situations.

The chapter also explains the differences between the program execution levels "cyclic processing", "time-controlled processing" and "interrupt-driven processing" and which blocks are available for your user program.

Chapter 5

This informs you about errors to be avoided when planning and writing your STEP 5 programs.
The chapter tells you about the help you can obtain from the system program for diagnosing errors and which reactions can be expected and informs you about the blocks in which you can program reactions to certain errors.
The chapter also explains the CPU 948R self-test.

Chapter 6	This covers the special functions integrated in the system program. It tells you how to use the special functions and how to call and assign parameters to the special function OBs. The chapter also explains how to recognize and deal with errors in the processing of a special function.
Chapter 7	This describes the use of data block DX 0 and its structure. The chapter informs you of the significance of the various DX 0 parameters. Based on examples, you will learn how to create data block DX 0 or how to assign the parameters in a screen form.
Chapter 8	This is a reference section for experienced system users. It provides information about the memory organization of the CPU 948R and certain system data words which contain information that can be called up by the user.
Chapter 9	This is also for experienced system users. The chapter explains how to address data in certain memory areas using absolute addresses.
Chapter 10	This explains how to connect your CPU to a PG and the functions provided by the PG software for testing your STEP 5 program
Chapter 11	This contains the Appendix with technical data (typical execution times for operations, system times, memory capacity) of the CPU 946R/947R and the CPU 948R.
Chapter 12	This lists documentation for further reading
Chapter 13	This is intended to help you find topics quickly and contains a list of abbreviations and a list of keywords as well as lists of all the numbered tables and figures.

Conventions Used in the Text

To provide you with an overview of the contents of the pages, the manual uses the following conventions in addition to a 2nd and 3rd order of titles:

Entries in the margin

Entries in the margin are keywords printed in italics on the left-hand edge of a page. They provide information about the contents of one or more paragraphs on the page.

Fourth order entries

Fourth order entries are not numbered but appear in the margin in bold face and identify a longer section of text.

The following conventions are also used.

Notes

Note
Important information is indicated in this format.

Instructions

Instructions (often a sequence of operations to be performed) are represented in tables, e.g.

Step	Action	Result
1	Switch the mode selector from RUN to STOP.	The CPU is in the stop mode. The STOP LED is lit continuously.
2	Hold the reset switch in the OVERALL RESET position; at the same time, switch the mode selector from STOP to RUN and back to STOP.	An OVERALL RESET is requested. The STOP LED flashes quickly.

Reference tables

Specific information you may require at any time is contained in numbered tables as shown in the following example and can be found in the list of tables (refer to Chapter 13).

Table 3-2 Binary logic operations

Operation	Operand	Function
A		AND logic operation with scan for signal state "1"
O		OR logic operation with scan for signal state "1"
	I 0.0 to 127.7	of an input in the PII
	.....	.....

Examples

Examples, some of which cover several pages, are highlighted by a gray frame. When the examples cover more than one page this is clearly indicated.

```
Example 1:   Calling and assigning parameters to a function block in the
             methods of representation STL and LAD/CSF in a program block

Method of representation STL

......
```

Contents

1 Introduction. . . . . 1 - 3

1.1 Typical Mode of Operation of a CPU . . . . . 1 - 4

1.2 The Programs in a CPU . . . . . 1 - 6

1.3 Which Operands are Available to the User Program? . . . . . 1 - 10

1.4 How Much Memory is Available for the User Program? . . . . . 1 - 13

1.5 How to Tackle Programming . . . . . 1 - 14

1.6 What You Need to Know to Program an H System . . . . . 1 - 17

1.7 Programming Tools . . . . . 1 - 18

1.8 What is New with the CPU 948R? . . . . . 1 - 19

2 User Program . . . . . 2 - 3

2.1 STEP 5 Programming Language . . . . . 2 - 4

2.1.1 The LAD, CSF, STL Methods of Representation . . . . . 2 - 4
2.1.2 Structured Programming . . . . . 2 - 5
2.1.3 STEP 5 Operations . . . . . 2 - 6
2.1.4 Number Representation . . . . . 2 - 8
2.1.5 STEP 5 Blocks and Storing Them in Memory . . . . . 2 - 12

2.2 Program, Organization and Sequence Blocks . . . . . 2 - 16

2.2.1 Organization Blocks as User Interfaces . . . . . 2 - 18
2.2.2 Organization Blocks for Special Functions . . . . . 2 - 22

2.3 Function Blocks . . . . . 2 - 23

2.3.1 Structure of Function Blocks . . . . . 2 - 24
2.3.2 Programming Function Blocks . . . . . 2 - 26
2.3.3 Calling Function Blocks and Assigning Parameters to Them . . . . . 2 - 28
2.3.4 Special Function Blocks . . . . . 2 - 33

2.4 Data Blocks . . . . . 2 - 35

2.4.1 Creating Data Blocks . . . . . 2 - 37
2.4.2 Opening Data Blocks . . . . . 2 - 38
2.4.3 Special Data Blocks . . . . . 2 - 41

3 Program Execution ... 3 - 3

3.1 Principle of Program Execution ... 3 - 4

3.2 Program Organization ... 3 - 5

3.3 Storing Program and Data Blocks ... 3 - 10

3.4 Processing the User Program ... 3 - 11

3.4.1 Definition of Terms used in Program Execution ... 3 - 12

3.5 STEP 5 Operations with Examples ... 3 - 15

3.5.1 Basic Operations ... 3 - 19

3.5.2 Programming Examples in the STL, LAD and CSF Methods of Representation ... 3 - 35

3.5.3 Supplementary Operations ... 3 - 51

3.5.4 Executive Operations ... 3 - 61

4 Operating Statuses and Program Execution Levels ... 4 - 3

4.1 Introduction and Overview ... 4 - 4

4.2 Program Execution Levels ... 4 - 7

4.3 STOP Mode ... 4 - 12

4.3.1 SOFT STOP ... 4 - 12

4.3.2 HARD STOP ... 4 - 16

4.3.3 OVERALL RESET ... 4 - 17

4.4 START-UP Mode ... 4 - 19

4.4.1 MANUAL and AUTOMATIC COLD RESTART ... 4 - 21

4.4.2 MANUAL and AUTOMATIC COLD RESTART WITH MEMORY ... 4 - 22

4.4.3 Comparison between COLD RESTART and COLD RESTART WITH MEMORY ... 4 - 25

4.4.4 User Interfaces for Start-Up ... 4 - 26

4.4.5 Interruptions during START-UP ... 4 - 27

4.4.6 Examples of Programs for START-UP ... 4 - 29

4.5 RUN Mode ... 4 - 31

4.5.1 Cyclic Program Execution ... 4 - 33

4.5.2 Specifying Time and Interrupt-Driven Program Execution ... 4 - 36

4.5.3 Time-Controlled Program Execution ... 4 - 37

4.5.4 Interrupt-Driven Program Execution ... 4 - 45

5 Interrupt and Error Diagnostics ... 5 - 3

5.1 Frequent Errors in the User Program ... 5 - 4

5.2 Error Information ... 5 - 5

5.3 Procedure for Error Analysis ... 5 - 9

5.4 Control Bits and Interrupt Stack ... 5 - 10
5.4.1 Control Bits ... 5 - 11
5.4.2 ISTACK Content ... 5 - 15
5.4.3 Example of Error Diagnosis using the ISTACK ... 5 - 20

5.5 Error Data Block (Error DB) ... 5 - 21
5.5.1 What is Entered in the Error DB? ... 5 - 21
5.5.2 Structure of the Error DB ... 5 - 22
5.5.3 Calling a User Interface ... 5 - 26

5.6 H Flag Word ... 5 - 27

5.7 Error Handling Using Organization Blocks ... 5 - 29

5.8 Causes of Error and Reactions of the CPU ... 5 - 32
5.8.1 OB 19: Calling a Logic Block That Is Not Loaded (KB) ... 5 - 33
5.8.2 OB 19: Opening a Data Block That Is Not Loaded (KDB) ... 5 - 33
5.8.3 OB 23/24, OB 28/29: Timeout Error (QVZ) ... 5 - 34
5.8.4 OB 25: Addressing Error (ADF) ... 5 - 35
5.8.5 OB 26: Cycle Time Exceeded Error (ZYK) ... 5 - 36
5.8.6 OB 27: (Substitution Error SUF) ... 5 - 37
5.8.7 OB 30: Parity Error and Timeout Error in the User Memory (PARE) ... 5 - 37
5.8.8 OB 32: Load and Transfer Error (TRAF) ... 5 - 38
5.8.9 OB 33: Collision of Timed Interrupts Error (WEFES/WEFEH) ... 5 - 39
5.8.10 OB 34: Error with G DB/GX DX (FEDBX) ... 5 - 41
5.8.11 OB 37: Entry in the Error DB ... 5 - 41

5.9 Self-Test ... 5 - 43
5.9.1 Overview ... 5 - 43
5.9.2 Organization of the Self-Test ... 5 - 44

6 Integrated Special Functions ... 6 - 3

6.1 Introduction ... 6 - 4
6.1.1 Interfaces ... 6 - 5
6.1.2 Error Handling ... 6 - 6
6.1.3 Notes on the Special Functions OB 124, OB 125 and OB 254/255 ... 6 - 7

6.2 OB 121: Set/Read System Time ... 6 - 8

6.3 OB 122: "Disable Interrupts" On/Off ... 6 - 12

6.4 OB 124: Delete STEP 5 Blocks ... 6 - 14

6.5 OB 125: Generate STEP 5 Blocks ... 6 - 17

6.6 OB 131: Delete ACCUs 1, 2, 3 and 4 ... 6 - 20

6.7 OB 132/133: Roll-Up ACCU/Roll-Down ACCU ... 6 - 21

6.8 OB 141: "Disable Single Cyclic Timed Interrupts" On/Off ... 6 - 23

6.9 OB 142: "Delay All Interrupts" On/Off ... 6 - 26

6.10 OB 143: "Delay Single Cyclic Timed Interrupts" On/Off ... 6 - 29

6.11 OB 150: Set/Read System Time ..... 6 - 32

6.12 OB 151: Set/Read Time for Real Time Interrupt ..... 6 - 37

6.13 OB 153: Set/Read Time for Delayed Interrupt ..... 6 - 44

6.14 OB 222: Restart Cycle Monitoring Time ..... 6 - 47

6.15 OB 254/255: Copy/Duplicate Data Blocks ..... 6 - 48

7 Extended Data Block DX 0 ..... 7 - 3

7.1 Application ..... 7 - 4

7.2 Parameter Assignment using the PG Screen Form ..... 7 - 5

7.3 Meaning of Some Parameters for Running the Program ..... 7 - 7

7.3.1 Updating the Timers ..... 7 - 7
7.3.2 Setting the Cycle Monitoring Times ..... 7 - 7
7.3.3 Level Priorities ..... 7 - 7

8 Memory Assignment and Memory Organization ..... 8 - 3

8.1 Structure of the Memory Area ..... 8 - 4

8.2 Memory Assignment in the CPU 948R ..... 8 - 5

8.2.1 Memory Assignment for the System RAM ..... 8 - 6
8.2.2 Memory Assignment for the I/O ..... 8 - 8

8.3 User Memory Organization in the CPU 948R ..... 8 - 10

8.3.1 Block Headers in User Memory ..... 8 - 12
8.3.2 Block Address List in Data Block DB 0 ..... 8 - 13
8.3.3 RI/RJ Area ..... 8 - 14
8.3.4 RS/RT Area ..... 8 - 15
8.3.5 Bit Assignment of the System Data Words ..... 8 - 18
8.3.6 Addressable System Data Area ..... 8 - 33

9 Memory Access Using Absolute Addresses ..... 9 - 3

9.1 Introduction ..... 9 - 4

9.2 Memory Access via Address in ACCU 1 ..... 9 - 8

9.2.1 LIR/TIR: Loading to or Transferring from a 16-Bit Memory Area Indirectly ..... 9 - 9
9.2.2 Examples of Access to DW > 255 ..... 9 - 15
9.2.3 LDI/TDI: Loading to or Transferring from a 32-Bit Memory Area Indirectly ..... 9 - 17

9.3 Transferring Memory Blocks ..... 9 - 19

9.4 Operations with the Base Address Register (BR Register) ..... 9 - 22

9.4.1 Operations for Transfer between Registers ..... 9 - 23
9.4.2 Accessing the Local Memory ..... 9 - 24
9.4.3 Accessing the Global Memory ..... 9 - 25
9.4.4 Accessing the Page Memory ..... 9 - 29

10 PG Interfaces and Functions ..... 10 - 3

10.1 Overview ..... 10 - 4

10.2 Notes on Connecting the PG and the PLC Modes ..... 10 - 5
10.2.1 Connecting the PG ..... 10 - 5
10.2.2 PLC Modes ..... 10 - 6

10.3 PG Functions ..... 10 - 8
10.3.1 Info ..... 10 - 9
10.3.2 Installation ..... 10 - 10
10.3.3 Program Test ..... 10 - 11

10.4 Serial Link PG - PLC via the Serial Interface ..... 10 - 18

10.5 PG Functions via the S5 Bus ..... 10 - 19
10.5.1 Application ..... 10 - 19
10.5.2 How the PG Functions via the S5 Bus Work ..... 10 - 20
10.5.3 General Notes on Installation ..... 10 - 21
10.5.4 Installation with One-Sided Operation ..... 10 - 22
10.5.5 Installation for Switched Operation ..... 10 - 26
10.5.6 Displaying Problems ..... 10 - 31

11 Appendix ..... 11 - 3

12 Further Reading ..... 12 - 3

13 Indexes ..... 13 - 3

List of Abbreviations ..... 13 - 3

List of Keywords ..... 13 - 5

List of Tables and Figures ..... 13 - 9
List of Tables ..... 13 - 9
List of Figures ..... 13 - 15

Introduction

1

Contents of Chapter 1

1.1 Typical Mode of Operation of a CPU ..... 1 - 4

1.2 The Programs in a CPU ..... 1 - 6

System Program ..... 1 - 6
User Program ..... 1 - 8

1.3 Which Operands are Available to the User Program? ..... 1 - 10

1.4 How Much Memory is Available for the User Program? ..... 1 - 13

1.5 How to Tackle Programming ..... 1 - 14

1.6 What You Need to Know to Program an H System ..... 1 - 17

1.7 Programming Tools ..... 1 - 18

1.8 What is New with the CPU 948R? ..... 1 - 19

Introduction

1

Aims of the manual

This manual is intended to provide specialized information about programming the CPU 948R for users who already have basic knowledge of programming PLCs and want to use the CPU 948R in the S5-155H programmable controller. If you do not yet have this basic knowledge, we strongly advise you read the documentation introducing the programming language STEP 5 /3/ or take part in a course at our training center. SIEMENS provides comprehensive training for SIMATIC S5. For more detailed information, contact your local SIEMENS office.

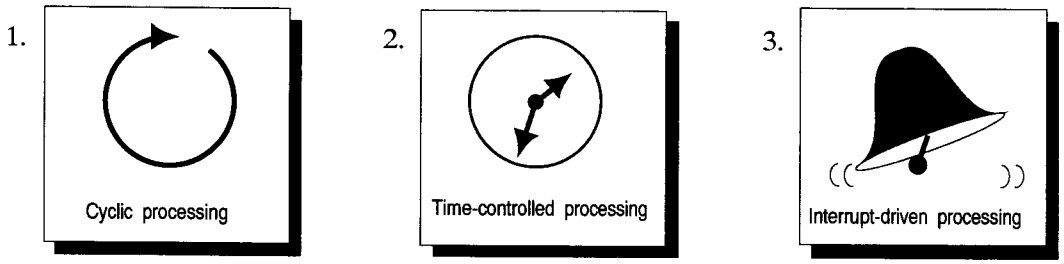
Contents of Chapter 1

Chapter 1 explains the typical mode of operation of a CPU and the structure of the CPU program.
You will also find a few suggestions about how to tackle programming and will learn some of the features of the CPU 948R which are important for programming.
If you have already worked with the CPU 946R/947R and would like to know the differences between these modules and the CPU 948R, refer to Section 1.8.

1.1 Typical Mode of Operation of a CPU

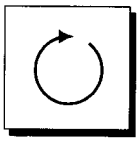
Mode of operation of a CPU

The following modes of operation are possible in a CPU:



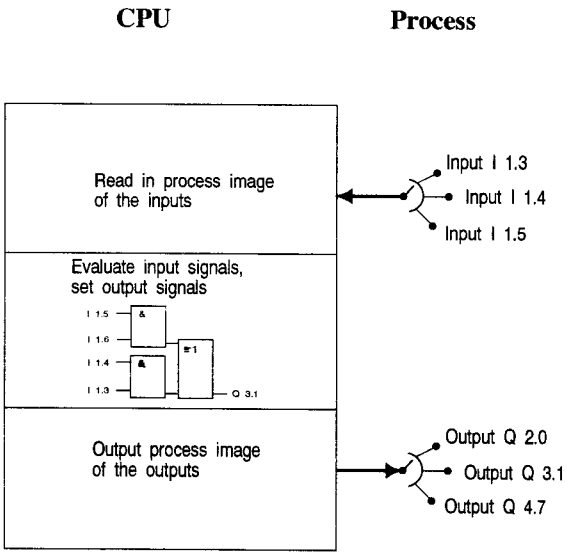
Cyclic processing

This is the main part of all activities in the CPU. As the name already says, the same operations are repeated in an endless cycle.



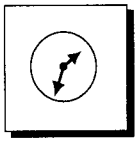
Cyclic processing can be divided into three main phases, as follows:

Phase	Sequence
1	All the input modules assigned to the CPU are scanned by the system program and the values read in are stored in the process image of the inputs (PII).
2	The values contained in the PII are processed by the user program and the values to be output are entered in the process image of the outputs (PIQ).
3	The values contained in the process image of the outputs are output by the system program to the output modules assigned to the CPU.



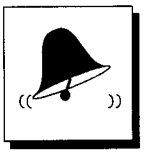
Time-controlled processing

In addition to cyclic processing, **time-controlled** processing is also available for processes requiring control signals at constant intervals, e.g. closed-loop control functions.



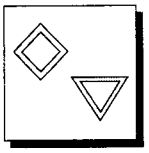
Interrupt-driven processing

If the reaction to a particular process signal must be particularly fast, this should be handled with **interrupt-driven** processing. With, for example, a system interrupt, triggered via an interrupt generating module, you can activate a special processing section within your program.



Processing according to priority

The types of processing listed above are handled by the CPU according to their **priority**.



Since a fast reaction is required to a time or interrupt event, the CPU interrupts cyclic processing to handle a time or interrupt event. Cyclic processing therefore has the lowest priority.

Whether or not the time-controlled processing is more important than the interrupt-driven processing depends, among other things, on the particular task. For this reason, the priority of time and interrupt-driven processing on the CPU 948R can be selected.

1.2 The Programs in a CPU

The program existing on every CPU is divided into the following:

- the **system program**

  and

- the **user program.**

System program

The system program organizes all the functions and sequences of the CPU which do not involve a specific control task (refer to Fig. 1-1).

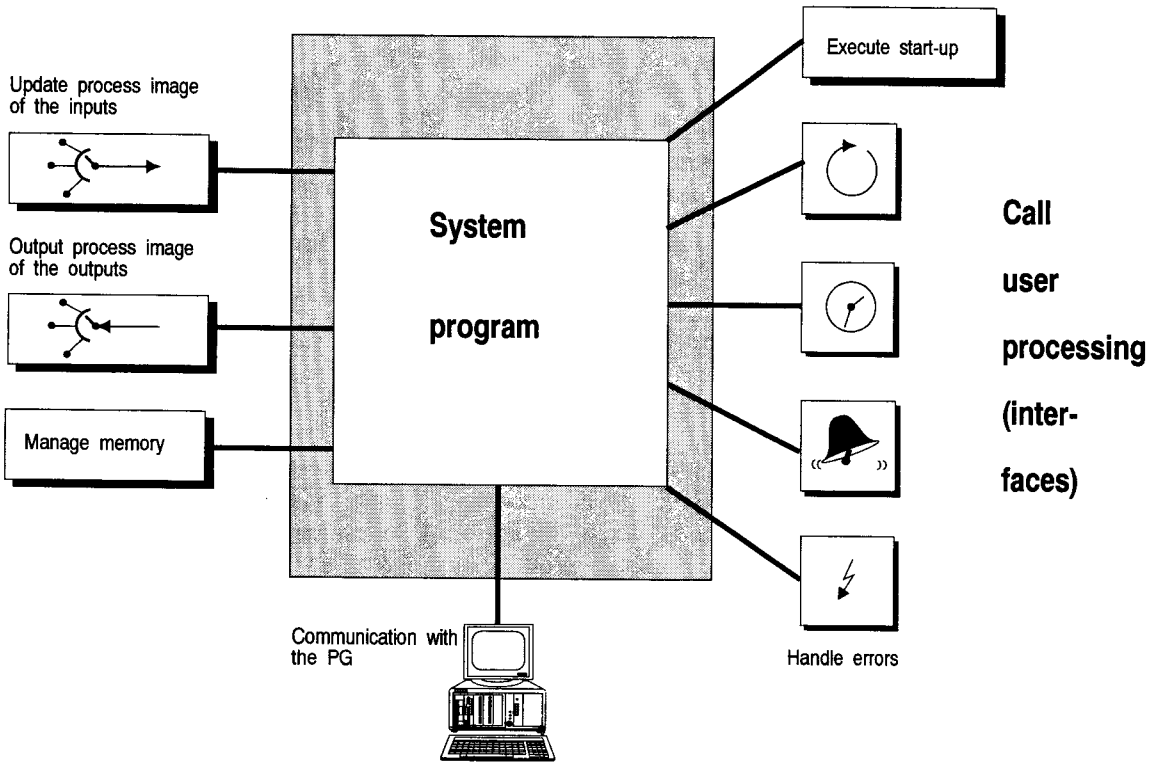


Fig. 1-1 Tasks of the system program

Tasks

The tasks include the following:

- restart (cold restart and cold restart with memory,
- updating the process image of the inputs and outputting the process image of the outputs,
- calling the cyclic, time-controlled and interrupt-driven programs,
- detection and handling of errors,
- memory management,
- communication with the programmer (PG).

User interfaces

As the user, you can influence the reaction of the CPU to particular situations and errors via special interfaces to the system program.

Storing the system program

After switching on the power supply to the PLC (POWER UP) the system program is read from the EPROM to the internal operating system RAM.

System program defaults

The following chapters, except for Chapter 7, describe the **default system reaction** to process events or errors. Depending on the defaults, the CPU changes to the stop mode if an operation code error occurs and the error organization block is not loaded.

Modifying the defaults

You can modify the system response by assigning parameters for the data block DX 0.
Chapter 7 describes the system response **following modification.**

User program

Tasks

The user program contains all the functions required for processing a **specific control task.** In general terms, these functions can be assigned to the interface provided by the system program for the various types of processing, as follows:

Type of processing	Task
Restart (cold restart and cold restart with memory)	To provide the conditions under which the other processing functions can start from a defined status following a cold restart of the controller (e.g. assigning specific values to signals).
Cyclic processing	Constantly repeated signal processing (e.g. logic operations on binary signals, reading in and analyzing analog values, specifying binary signals for output, outputting analog values).
Time-controlled processing	Special, time-dependent processing with the following time conditions: - faster than the average cycle, - at a time interval greater than the average cycle time, - at a specified point in time.
Interrupt-driven processing	Special, fast reactions to certain process signals.
Error reaction	Handling problems within the normal sequence of the program.

Structure

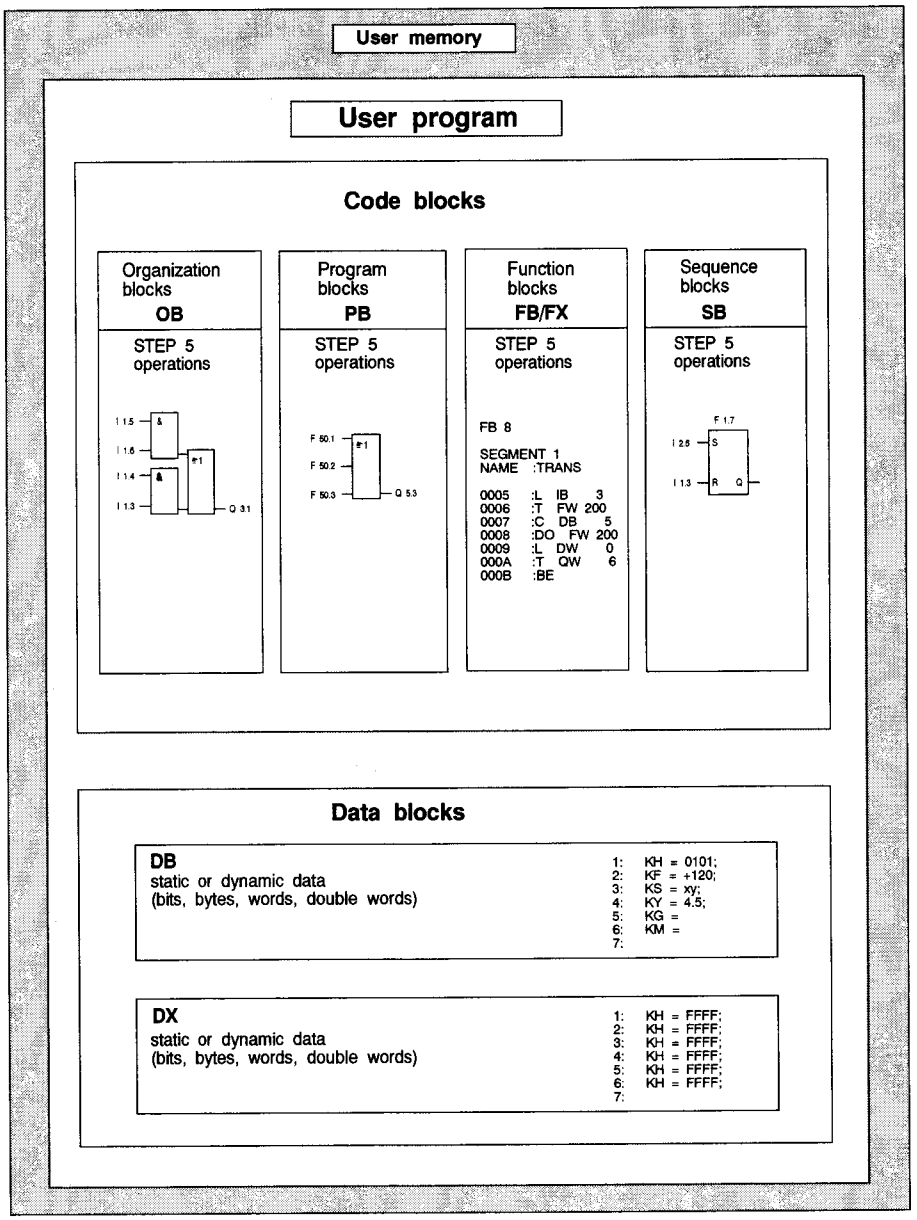


Fig. 1-2 Structure of a STEP 5 user program

Storing the user program

After programming the user program, you must load it in the user memory of the CPU 948R (directly from the PG) or via a memory card whose contents are copied to the user memory by an OVERALL RESET of the CPU.

Interfaces to the system program

Organization blocks are available as interfaces to the system program for the special types of processing.

1.3 Which Operands are Available to the User Program?

The CPU 948R provides the following operand areas for programming:

- process image and I/Os
- flags (F flags and S flags)
- timers/counters
- data blocks (data words)

Process image of the inputs and outputs PII/PIQ

Characteristics	Size
The user program can access the following data types in the process image extremely quickly: - single bits, - bytes, - words, - double words	128 bytes each for inputs and outputs

I/O area (P area)

Characteristics	Size
The user program can access the I/O modules directly via the S5 bus. The following data types are possible: - bytes, - words.	256 bytes each for inputs and outputs

Extended I/O area (O area)

Characteristics	Size
The user program can access the I/O modules directly via the S5 bus. The following data types are possible: - bytes, - words.	256 bytes each for inputs and outputs

F flags

Characteristics	Size
The flag area is a memory area which the user program can access extremely quickly with certain operations. The flag area should be used ideally for working data required often. The following data types can be accessed: - single bits, - bytes, - words, - double words. Single flag bytes can be used as **interprocessor communication flags** (IPC flags) to exchange data between the CPU and CPs/IPs. IPC flags are updated by the system program at the end of the cycle via a buffer in the CP/IP.	2048 bits

S flags (extended flag area)

Characteristics	Size
The CPU 948R also contains an additional flag area, the S flag area. The user program can also access this area extremely quickly as with the F flags. S flags **cannot**, however, be used as **actual operands** with function block calls nor as **IPC flags**. The bit test operations of the CPU 948R can also not be used with the S flags. These flags can only be used with the PG system software "S5-DOS" from version 3.0 upwards or "S5-DOS/MT" from version 1.0 upwards.	32 768 bits

Timers (T)

Characteristics	Size
The user program loads timer cells with a time value between 10 ms and 9990 s and by means of a start operation, decrements the timer from this value at the preselected intervals until it reaches the value zero.	256 timer cells

Counters (C)

Characteristics	Size
The user program loads counter cells with a start value (max. 999) and then increments or decrements them.	256 counters

Data words in the current data block

Characteristics	Size
A data block contains constants and/or variables in the byte, word or double word format. With STEP 5 operations, you can always access the "current" data block (refer to Section 2.4.2). The following data types can be accessed: - single bits, - bytes, - words, - double words.	256 words 1)

1) In data blocks with a length greater than 256 words, you can only access data words with the numbers > 255 with operations for absolute memory access (refer to Chapter 9).

1.4 How Much Memory is Available for the User Program?

For storing logic and data blocks, the CPU 948R only has the user memory in the internal RAM.

The CPU 948R is available with two versions of the user memory:

- Version 1: with 640 Kbytes,
- Version 2: with 1,664 Kbytes.

1.5 How to Tackle Programming

If you are an experienced user, you have probably found the most suitable method for creating programs for yourself and you can skip this section.

Less experienced readers will find tips for designing, programming, testing and starting up your STEP 5 program.

Implementation stages

The implementation of the STEP 5 control program can be divided into three stages:

Stage	Activity
1	Determining the technological task
2	Designing the program
3	Creating, testing and starting the program

Recursive procedure

In practice, you will recognize that certain steps must be repeated (recursive procedure), e.g. when you realize that more signals are required to improve the handling of the task.

Stage 1

Determining the technological task:

Stage	Activity
1	Create a general block diagram outlining the control tasks of your process.
2	Create a list of the input and output signals required for the task.
3	Improve the block diagram by assigning the signals and any particular time conditions and/or counter statuses to the individual blocks.

Stage 2

Designing the program

Stage	Activity
1	Based on the improved block diagram, decide on the types of processing required of your program (cyclic processing, time-controlled processing etc.) and select the OBs required for this.
2	Divide the types of processing into technological and/or functional units.
3	Check whether the units can be assigned to a program or function block and select the blocks you require (PB x, FB y etc.)
4	Find out which timers, counters and data or results memory you require.
5	Specify the tasks for each of the proposed logic blocks and the data for flags and data blocks which may be required. Create flow diagrams for the logic blocks.

Notes on the scope of cyclic processing

When deciding on the types of processing, keep the following conditions in mind:

- The cycle must run through quickly enough. The process statuses must not change more quickly than the CPU can react. Otherwise the process can get out of control.

- The maximum reaction time should be taken as twice the cycle time.
  The cycle time is determined by the cyclic processing of the system program and the type and scope of the user program. It is often not constant, since the cyclic user program may be interrupted when time and interrupt-driven program sections are called.

Stage 3

Creating, testing and starting up the program:

Stage	Activity
1	Decide on the type of representation for the logic blocks (LAD, CSF or STL, refer to Chapter 2). Remember that function blocks can only be created in the STL method of representation.
2	Program all logic and data blocks (please refer to your STEP 5 manual).
3	In contrast to the CPU 948, configure the structure of the I/Os and your transfer data (data block lists) with COM 155H (see Volume 1/1).
4	Start up the blocks one after the other (you may have to program a different OB 1 for each individual step, to call the logic blocks): 1a: load the block(s) 1b: test the block(s) (For more detailed information please refer to your STEP 5 manual and Chapter 10).
5	When you are certain that all the logic blocks run correctly and all the data can be correctly calculated and stored, you can start up your whole program.

Note on test strategies

When you actually start up your program for the first time in genuine process operation, i.e. with real input and more importantly output signals, is a decision that must be left up to yourself or to a team of experts.
The more complex the process, the greater the risk and therefore the greater the care required when starting up.

1.6 What You Need to Know to Program an H System

Range of operations

Programming the CPU 948R is essentially the same as programming a CPU 948. With the exception of the operations for the multiprocessor mode, you can use all the STEP 5 operations.

PG operation

Operating the programmer (PG) is exactly the same as with the S5-155U. The PG is only connected to one of the two central controllers (CC); the data you input are automatically transferred to the second CC when the S5-155H is operating in the redundant mode.

Program execution

While the user program is running on the S5-155H, alongside the cyclic processing of the process images on the CPUs of the subunits other system functions are executed for synchronization, for exchanging and comparing data and, in some situations, operating modes.

Event-synchronized user program execution

The same user program is run in both devices (master and reserve) of the S5-155H. The master and reserve operate **event synchronized**. The system program of the CPU 948R makes sure that both devices work with identical data (for more information refer to Part I of Volume 1 of this manual).

Process image of the inputs

At the start of each cycle, the master and reserve read the process images of the one-sided and redundant inputs assigned to them. The process image of the switched inputs is only read in by the master. The two devices then exchange the complete process image of the inputs (for more information refer to Part I of Volume 1 of this manual).

1.7 Programming Tools

Suitable PGs

The following programmers are available for creating your user program, PG 685, PG 710, PG 730, PG 750 and PG 770. You can check on the performance and characteristics of these devices in the catalog ST 59 /9/.

> **Note**
> If you wish to use the **full range of performance** of the CPU 948R in your automation software, you require the PG system software "**STEP 5/ST**" from version 6.3 upwards or "STEP 5/MT" from version 6.0 upwards plus the "Delta diskette CPU 948" and a PG 7xx.

Suitable software

You can create **user programs for SIMATIC S5 programmable controllers** as follows:

- In the **STEP 5** programming language,

  Here you require the STEP 5 programming package along with the system software STEP 5/ST or STEP 5/MT (description, refer to /3/ in Further Reading),

  or

- In a higher programming language:

  If you are familiar with programming in higher programming languages, you can also formulate your STEP 5 program for the CPU 948R as follows:

  - **SCL** (refer to /12/ in Further Reading, the SCL compiler is contained in the PG software "S5-DOS/MT" from version 6 upwards.)

    or

  - **C with S5 C compiler** (refer to /13/ in Further Reading).

You can also create **programs for sequence control systems** in a graphic representation using the **GRAPH 5** programming package (description, refer to /4/ in Further Reading).

Depending on the task, you can also incorporate "off-the-peg" standard function blocks in your user program. The performance and characteristics of these blocks are described in the catalog ST 57 /11/.

1.8 What is New with the CPU 948R?

Compared with the CPU 946R/947R, the CPU 948R has new characteristics and functions and significant improvements in performance.

Faster processing

The CPU 948R is between three and five times faster than the CPU 946R/947R. The exact execution times can be found in your Pocket Guide /1/.

Larger user memory

CPU 948R only RAM (integrated)	CPU 946R/947R RAM or EPROM cartridges
640 Kbytes or 1664 Kbytes in internal RAM integrated in the CPU 948R, for logic and data blocks.	Max. 896 Kbytes, of which 128 Kbytes fixed as RAM plus additional 6 RAM or EPROM cartridges, each with 128 Kbytes in 355 memory modules.

Memory Card

CPU 948R	CPU 946R/947R
SIMATIC Memory Card (Flash EPROM), 256 Kbytes to 2 Mbytes; the memory card is programmed on the PG and its contents loaded in the user memory (internal RAM) following an OVERALL RESET.	SIMATIC memory cartridges RAM or EPROM, max. 128 Kbytes; data blocks can be copied to the DB RAM from EPROM (OB 254/255).

Expansion of time-controlled program processing

Apart from the familiar cyclic timed interrupts at fixed intervals, the following new interrupts are also processed:

- delayed interrupt (OB 6),
- real time interrupt (OB 9).

Both interrupts are only possible when "process interrupts via IB 0 = **off**" is set in DX 0.

Expansion of PG communication

- PG functions via SINEC H1 and parallel backplane bus.

New special functions

- Delete ACCU 1 to ACCU 4 (OB 131)
  Roll-up ACCU (OB 132)
  Roll-down ACCU (OB 133)
- Enable/disable "block single timed interrupts" (OB 141)
  Enable/disable "delay all interrupts" (OB 142)
  Enable/disable "delay single timed interrupts" (OB 143)
- Read/set system time, compatible with CPU 928B (OB 150),
- Assign parameters for real time interrupt (OB 9), (OB 151)
- Assign parameters for delayed interrupt (OB 6), (OB 153)
- Restart cycle monitoring time (OB 222)

Larger BSTACK

The BSTACK of the CPU 948R can contain 20 entries more (max. 60) than the BSTACK of the CPU 946R/947R.

User Program

2

Contents of Chapter 2

2.1	STEP 5 Programming Language	2 - 4
2.1.1	The LAD, CSF, STL Methods of Representation	2 - 4
2.1.2	Structured Programming	2 - 5
2.1.3	STEP 5 Operations	2 - 6
2.1.4	Number Representation	2 - 8
2.1.5	STEP 5 Blocks and Storing Them in Memory	2 - 12
2.2	Program, Organization and Sequence Blocks	2 - 16
2.2.1	Organization Blocks as User Interfaces	2 - 18
2.2.2	Organization Blocks for Special Functions	2 - 22
2.3	Function Blocks	2 - 23
2.3.1	Structure of Function Blocks	2 - 24
2.3.2	Programming Function Blocks	2 - 26
2.3.3	Calling Function Blocks and Assigning Parameters to Them	2 - 28
2.3.4	Special Function Blocks	2 - 33
2.4	Data Blocks	2 - 35
2.4.1	Creating Data Blocks	2 - 37
2.4.2	Opening Data Blocks	2 - 38
2.4.3	Special Data Blocks	2 - 41

User Program

2

The following chapter explains the components that make up a STEP 5 user program for the CPU 948R and how it can be structured.

2.1 STEP 5 Programming Language

With the STEP 5 programming language, you convert automation tasks into programs that run on SIMATIC S5 programmable controllers. You can program simple binary functions, complex digital functions and arithmetic operations including floating point arithmetic using STEP 5.

Types of operation

The **operations** of the STEP 5 programming language are divided into the following groups:

Basic operations

- you can use these operations in all logic blocks
- methods of representation: ladder diagram (LAD), control system flowchart (CSF), statement list (STL).

Supplementary operations and system operations:

- can only be used in function blocks
- only statement list (STL) method of representation
- system operations: only experienced STEP 5 programmers should use system operations

2.1.1 The LAD, CSF, STL Methods of Representation

When programming in STEP 5, you can choose between the three methods of representation ladder diagram (LAD), control system flowchart (CSF) and statement list (STL) for each individual logic block. You can choose the method of representation that best suits your particular application.

The machine code MC5 that the programmers (PGs) generate is the same for all three methods of representation.

If you follow certain rules when programming in STEP 5 (see /3/), the programmer can translate your user program from one method of representation into any other.

Graphic representation or list of statements

While the ladder diagram (LAD) and control system flowchart (CSF) methods of representation represent your STEP 5 program graphically, statement list (STL) represents STEP 5 operations individually as mnemonic abbreviations.

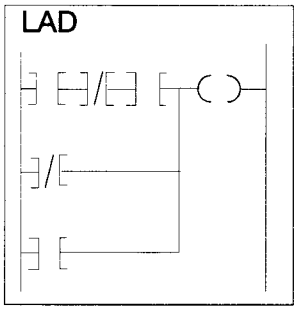
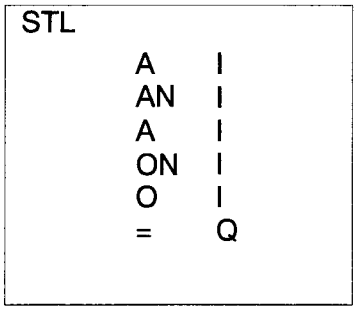
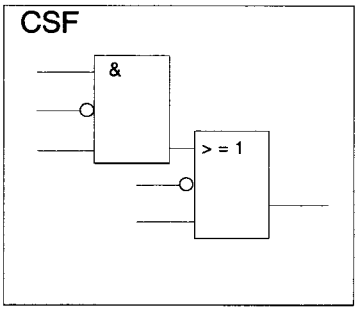
Ladder diagram	Statement list	Control system flowchart
Programming with graphic symbols like a circuit diagram	Programming with mnemonic abbreviations of function designations	Programming with graphic symbols
complies with DIN 19239	complies with DIN 19239	complies with IEC 117-15 DIN 40700 DIN 40719 DIN 19239
		

Fig. 2-1 Methods of representation in the STEP 5 programming language

Graphic representation of sequential controls

GRAPH 5 /4/ is a programming language for graphic representation of sequential controls. It is at a higher level than the LAD, CSF, STL methods of representation. A program written in GRAPH 5 as a graphic representation is automatically converted to a STEP 5 program by the PG.

2.1.2 Structured Programming

Using STEP 5, you can structure your program by dividing it into self-contained program sections (blocks). This division of your program clarifies the essential program structures making it easy to recognize the system parts that are related within the software.

Structured programming offers you the following advantages:

- simple and clear creation of programs, even large ones
- standardization of program parts
- simple program organization
- easy program changes
- simple, section by section program test
- simple system start-up

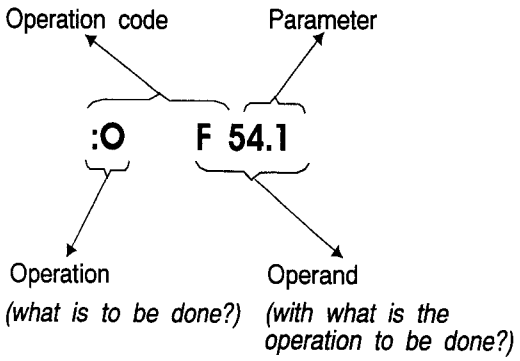
What is a block?

A block is a part of the user program that is distinguished by its function, structure or application. You can differentiate between blocks that contain **statements** (code) i.e. organization blocks, program blocks, function blocks or sequence blocks, and blocks that contain **data** (data blocks).

2.1.3 STEP 5 Operations

A STEP 5 operation is the smallest independent unit of the user program. It is the work specification for the CPU. A STEP 5 operation consists of an operation and an operand as shown in the following example:

Example



Absolute and symbolic operands

You can enter the operand **absolutely** or **symbolically** (using an assignment list) as shown in the following example:

Absolute representation: :A I 1.4

Symbolic representation: :A -Motor1

For more information on absolute and symbolic programming, refer to your STEP 5 manual.

Application of STEP 5 operations

The STEP 5 operation set enables you to do the following:

- set or reset and combine binary values logically
- load and transfer values
- compare values and process them arithmetically
- specify timer and counter values
- convert number representations
- call blocks and execute jumps within a block

  and

- influence program execution

Result of logic operation RLO

The central bit for controlling the program is the result of logic operation RLO. This is obtained as a result of binary logic operations and is influenced by some operations.

Section 3.5 describes the whole STEP 5 operation set and explains how the RLO is obtained. This section also includes programming examples for individual STEP 5 operations.

2.1.4 Number Representation

To allow the CPU to logically combine, modify or compare numerical values, these values must be located in the accumulators (working registers of the CPU) as binary numbers.

Depending on the operations to be carried out, the following number representations are permitted in STEP 5:

Binary numbers: 16-bit fixed point numbers

32-bit fixed point numbers

32-bit floating point numbers (with a 24-bit mantissa)

Decimal numbers: BCD-coded numbers (sign and 3 digits)

Numerical input on the PG

When you use a programmer to input or display number values, you set the data format on the programmer (e.g. KF or fixed point) in which you intend to enter or display the values. The programmer converts the internal representation into the form you have requested.

Permitted operations

You can carry out **all arithmetic operations** with the 16-bit fixed point numbers and floating point numbers, including comparison, addition, subtraction, multiplication and division.

> **Note**
> Do not use BCD-coded numbers for arithmetical operations, since this leads to incorrect results.

Use 32-bit fixed point numbers to execute comparison operations. These are also necessary as an intermediate level when converting numbers in BCD code to floating point numbers. With the operations +D and -D they can also be used for addition and subtraction.

The STEP 5 programming language also has **conversion operations** that enable you to convert numbers directly to the most important of the other numerical representations.

16-bit and 32-bit fixed point numbers

Fixed point numbers are whole binary numbers with a sign.

Coding of fixed point numbers

Fixed point numbers are 16 bit (= 1 word) or 32 bit (= 2 words) in binary representation. Bit 15 or bit 31 contains the sign.

- '0' = positive number
- '1' = negative number

The two's complement representation is used for negative numbers.

PG input

Input of 16-bit fixed point number data format at the PG:KF

Input of 32-bit fixed point number data format at the PG:DH

Permitted numerical range

-32768 to +32767 (16 bit)

-2147483648 to +2147483647 (32 bit)
(8000 0000H to 7FFF FFFFH)

Using fixed point numbers

Use fixed point numbers for simple calculations and for comparing number values. Since fixed point numbers are always whole numbers, remember that the result of dividing two fixed point numbers is also a fixed point number without decimal places.

Floating point numbers

Floating point numbers are positive and negative fractions. They always occupy a double word (32 bits). A floating point number is represented as an exponential number. The mantissa is 24 bits long and the exponent is 8 bits long.

In the CPU 948R, the default mantissa is 24 bits long (bits 0 to 23) for adding, subtracting, multiplying and dividing.

The exponent indicates the order of magnitude of the floating point number. The sign of the exponent tells you whether the value of the floating point number is greater or less than 0.1.

Using floating point numbers

Use floating point numbers for solving extensive calculations, especially for multiplication and division or when you are working with very large or very small numbers!

Accuracy

The mantissa indicates the accuracy of the floating point number as follows:

- Accuracy with a 24-bit mantissa:

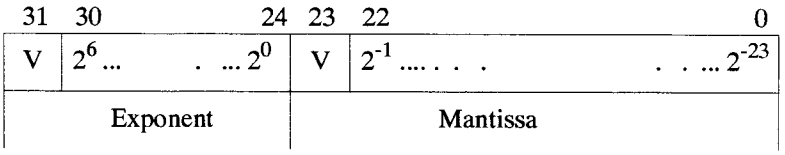
  2^{-24} = **0.000000059604** (corresponds to 7 decimal places)

If the sign of the mantissa is "0" the number is positive; if the sign is "1" it is a negative number in its two's complement representation.

The **floating point value '0'** is represented as the binary value **80000000H** (32 bits, see below).

Coding floating point numbers

Coding a floating point number:



Specification of the data format for floating point numbers at the PG: KG

Permissible numerical range

$\pm 0.1469368 \times 10^{-38}$ to $\pm 0.1701412 \times 10^{39}$

Input/output on PG

a) in a logic block:

You want to load the number N = 12.34567 as a floating point number.

Input:

:LKG1234567+2

PG display after you enter the line:

:L KG + 1234567 + 02

Mantissa with sign — Exponent (base 10) with sign

Value of the number input: $+0.1234567 \times 10^{+2} = 12.34567$

b) in a data block:

You want to define the number N = - 0.005 as a floating point constant.

Input:

6: KG = - 5 - 2

PG display after you enter the line:

6: KG =- 5000000 - 02

Mantissa with sign — Exponent (base 10) with sign

Value of the number input : $- 0.5 \times 10^{-2} = 0.005$

Numbers in BCD code

Decimal numbers are represented as numbers in BCD code. With their sign and three digits, they occupy 16 bits (1 word) in an accumulator as shown in the following example:

15 12	11 8	7 4	3 0
V V V V	hundreds	tens	ones

The individual digits are positive 4-bit binary numbers between 0000 and 1001 (0 and 9 decimal).

The left bits are reserved for the sign as follows:

Sign for a positive number:	0000
Sign for a negative number:	1111

Permissible numerical range

-999 to +999

2.1.5 STEP 5 Blocks and Storing Them in Memory

Identification

A block is identified as follows:

- the block type (OB, PB, SB, FB, FX, DB, DX)

  and

- the block number (number between 0 and 255).

Block types

The STEP 5 programming language differentiates between the following block types:

Organization blocks (OB)

Organization blocks are the interface between the system program and the user program. They can be divided into two groups as follows:

With OB 1 to OB 39, you can control program execution, the restart procedure of the CPU and the reaction in the event of an error. You program these blocks yourself according to your automation task. These OBs are called by the system program.

OBs 40 to 119 are blocks belonging to the operating system. You must not call these blocks.

OBs 121 to 255 contain special functions of the system program. You can call these blocks, if required, in your user program.

Program blocks (PB)

You require program blocks to structure your program. They contain program parts divided according to technological and functional criteria. Program blocks represent the heart of the user program.

Sequence blocks (SB)

Sequence blocks were originally special program blocks for step by step processing of sequencers. In the meantime, however, sequencers can be programmed with GRAPH 5 /4/. Sequence blocks have therefore lost their original significance in STEP 5.
Sequence blocks now represent an extension of the program blocks and are used as program blocks.

Function blocks (FB/FX)

You use function blocks to program frequently recurring and/or complex functions (e.g. digital functions, sequence control systems, closed loop controls and signalling functions).

A function block can be called several times by higher order blocks and supplied with new operands (assigned parameters) at each call. Using block type FX doubles the maximum number of possible function blocks.

Data blocks (DB/DX)

Data blocks contain the (fixed or variable) data with which the user program works. This type of block contains no STEP 5 statements and has a distinctly different function from the other blocks. Using block type DX doubles the number of possible data blocks.

Formal structure of the blocks

All blocks consist of the following two parts:

- a block header

  and

- a block body

Block header

The **block header** is always 5 words long and contains information for block management in the PG and data for the system program.

Block body

Depending on the block type, the **block body** contains the following:

- STEP 5 operations (in OB, PB, SB, FB, FX),
- variable or constant data (in DB, DX)

  and

- a formal operand list (in FB, FX).

Block preheader

The programmer also generates a **block preheader** (DV, DXV, FV, FXV) for block types DB, DX, FB and FX. These block preheaders contain information about the data format (for DB and DX) or the jump labels (for FB and FX). Only the PG can evaluate this information. Consequently the block preheaders are not transferred to the CPU memory. You cannot influence the contents of the block header directly.

Maximum length

A STEP 5 block can occupy a maximum of 32 767 words in the program memory of the CPU (1 word corresponds to 16 bits).

Available blocks

You can program the following block types:

OB	1 to 39	
FB	0 to 255	total 512
FX	0 to 255	
PB	0 to 255	
SB	0 to 255	
DB	2 to 255	total 508 [1]
DX	3 to 255	

Data blocks DB 0, DB 1, DX 0 and DX 1 contain parameters. These are reserved for specific functions and you cannot use them as normal data blocks.
Data block DX 2 is reserved for the 2nd serial interface and you should not use it.

1) Of these, two numbers from the 'DB' area must be left free for the error DB and RAM
DB.

Block storage

The PG stores all programmed blocks in the program memory in the order in which they are transferred (Fig. 2-2). With the PG function "transfer data blocks A" the logic blocks are transferred first followed by the data blocks.

The start addresses of all stored blocks are placed in an address list in data block DB 0.

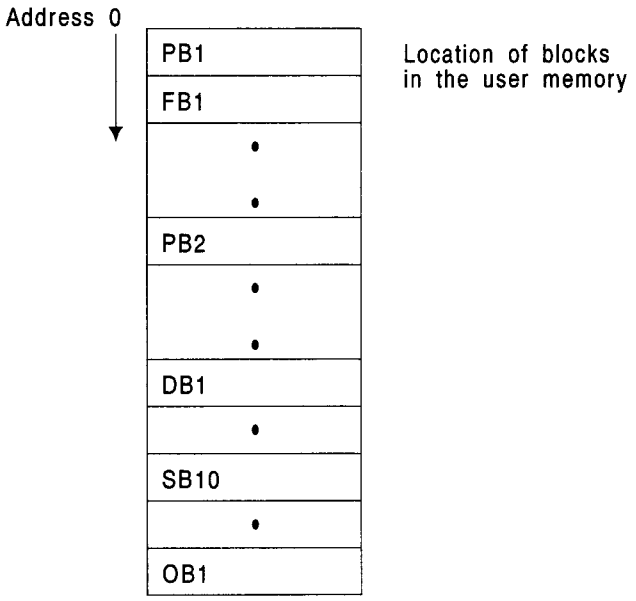


Fig. 2-2 Example of block storage in the user memory

Correcting and deleting blocks

When you **correct** blocks, the old block is declared invalid in the memory and a new block is entered.
Similarly, when blocks are **deleted**, they are not really deleted, instead they are declared invalid. The space they occupy is, however, not released and is not available for blocks loaded later.

> **Note**
> You can use the COMPRESS MEMORY online function to make space for new blocks. This function optimizes the utilization of the memory by deleting blocks marked as invalid and shifting valid blocks together.

2.2 Program, Organization and Sequence Blocks

Program blocks (PBs), organization blocks (OBs) and sequence blocks (SBs) are the same with respect to programming and calling. You can program all three types in the LAD, CSF and STL methods of representation.

Programming

When programming organization, program and sequence blocks, proceed as follows:

Step	Action
1	First indicate the type of block and then the number of the block that you want to program. The following numbers are available for the type of block listed: - program blocks 0 to 255 - sequence blocks 0 to 255 - organization blocks 1 to 39
2	Enter your program in the STEP 5 programming language. When programming PBs, OBs and SBs, you can only use the STEP 5 **basic** operations! A STEP 5 block should always be a self-contained program section. Logic operations must always be completed within a block.
3	Complete your program input with the block end operation "BE".

Block calls

With the exception of OB 1 to OB 39 you must call the blocks to process them. Use the special STEP 5 block call operations to call the blocks.
You can program block calls inside an organization, program, function or sequence block. They can be compared with jumps to a subroutine. Each jump causes a block change. The return address within the calling block is buffered by the system.

Block calls can be unconditional or conditional as follows:

Unconditional call

The "JU" statement belongs to the unconditional operations. It has no effect on the RLO. The RLO is carried along with the jump to the new block. Within the new block, it can be evaluated but no longer combined logically.

The addressed block is processed **regardless** of the previous result of logic operation (RLO - see Section 3.4).

Example: **JU** PB 100

Conditional call

The JC statement belongs to the conditional operations. The addressed block is processed only if the previous **RLO = 1**. If the RLO = 0, the jump is not executed.

Example: **JC** PB 100

> **Note**
> After the conditional jump operation is executed, the RLO is set to "**1**" regardless of whether or not the jump to the block is executed.

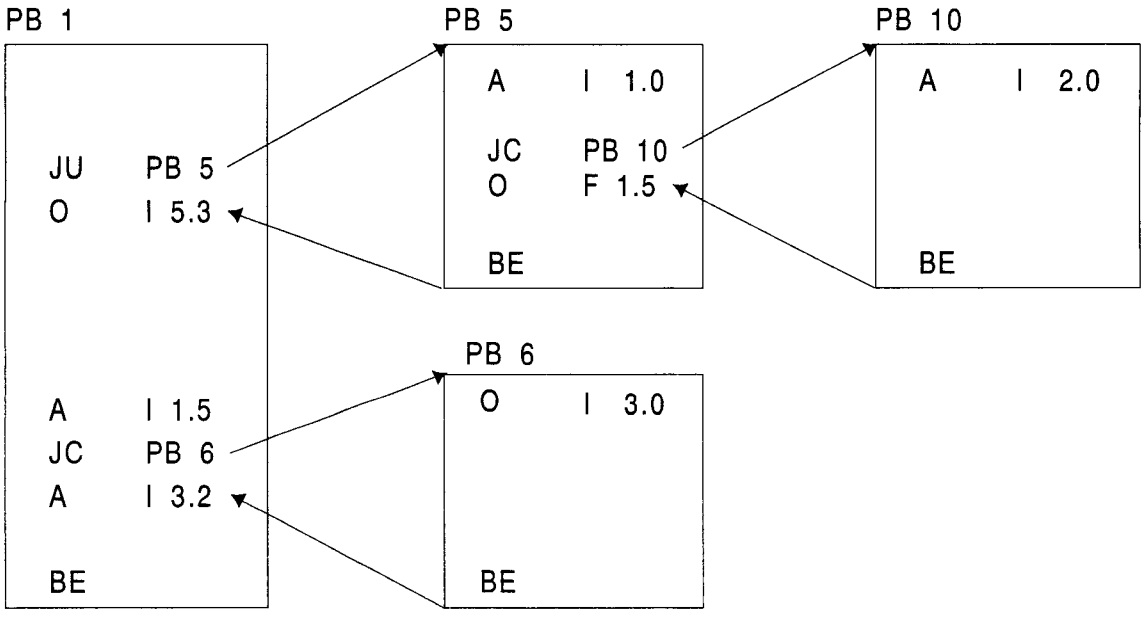


Fig. 2-3 Block calls that enable processing of a program block

Effect of the BE statement

After the "BE" statement (block end), the CPU continues the user program in **the block** in which the **block call** was programmed. Program execution continues at the STEP 5 statement following the block call.

The "BE" statement is executed regardless of the RLO. After "BE", the RLO can no longer be combined logically. However, the RLO or arithmetic result occurring directly before execution of the BE operation is transferred to the block where the call originated and can be evaluated there. When program execution returns from the block that has been called, the contents of ACCU 1, ACCU 2, ACCU 3 and ACCU 4, the condition codes CC 0 and CC 1 and the RLO are not changed. (Refer to Section 3.5 for more detailed information about the ACCUs, CC0/CC1 and RLO).

2.2.1 Organization Blocks as User Interfaces

Organization blocks form the interfaces between the system program and the user program. Organization blocks OB 1 to OB 39 belong to your user program just as program blocks. By programming these OBs, you can influence the behavior of the CPU during start-up, program execution and in the event of an error. The organization blocks are effective as soon as they are loaded in the PLC memory. **This is also possible while the PLC is in the run mode.**

Once the system program has called a specific organization block, the user program it contains is executed.

> **Note**
> You can program blocks OB 1 to OB 39 as user interfaces and they are called automatically by the system program as a reaction to certain events.
>
> For **test purposes**, you can also call these organization blocks from the user program (JC/JU OB xxx). It is, however, not possible to trigger a COLD RESTART, e.g. by calling OB 20.

The following table provides you with an overview of the user interfaces (OBs).

Table 2-1 Overview of the organization blocks of the CPU 948R for program execution

<table>
<tr><th colspan="4">Organization blocks for controlling program execution</th></tr>
<tr><th>Block</th><th colspan="3">Function and call criterion</th></tr>
<tr><td>OB 1</td><td colspan="3">Organization of cyclic program execution; first call after a start-up, then cyclic call.</td></tr>
<tr><td>OB 2
OB 3
OB 4
OB 5
OB 6
OB 7
OB 8
OB 9</td><td colspan="3">With DX-0 setting "Process interrupt servicing via input byte IB 0 =on":
(interruptability at block boundaries)

Call with signal state change in input byte IB 0 in bit:
I 0.0
I 0.1
I 0.2
I 0.3
I 0.4
I 0.5
I 0.6
I 0.7</td></tr>
<tr><td>OB 6
OB 9</td><td colspan="3">With DX-0 setting "Process interrupt servicing via input byte IB 0 = off":
(interruptability at block boundaries)

Delayed interrupt
Real time interrupt</td></tr>
<tr><td rowspan="2">OB 10
OB 11
OB 12
OB 13
OB 14
OB 15
OB 16
OB 17
OB 18</td><td colspan="3">Organization of time-controlled program execution (timed interrupt) with selectable basic clock rate (default T = 100 ms) and clock distributor (default corresponds to 150U) in data block DX 0; (interruptability at block boundaries)
Calls as follows:</td></tr>
<tr><td>
<table>
<tr><th>Default setting</th><th>150U clock distributor</th><th>2^n clock distributor</th></tr>
<tr><td>0.1 s
0.2 s
0.5 s
1.0 s
2.0 s
5.0 s
10.0 s
20.0 s
50.0 s</td><td>T * 1
T * 2
T * 5
T * 10
T * 20
T * 50
T * 100
T * 200
T * 500</td><td>T * 1
T * 2
T * 4
T * 8
T * 16
T * 32
T * 64
T * 128
T * 256</td></tr>
</table>
</td></tr>
</table>

Table 2-2 Overview of the organization blocks of the CPU 948R for start-up

Organization blocks to control the start-up procedure	
Block	**Function and call criterion**
OB 20	Call on request for COLD RESTART (manual and automatic)
OB 21	Call on request for MANUAL COLD RESTART WITH MEMORY
OB 22	Call on request for AUTOMATIC COLD RESTART WITH MEMORY

Table 2-3 Organization blocks of the CPU 948R for a SOFT STOP

Organization blocks to control the start-up procedure	
Block	**Function and call criterion**
OB 38	Organization of the start-up procedure for communication in the "soft stop" mode.
OB 39	Organization of the cyclic program for communication in the "soft stop" mode.

Table 2-4 Overview of the organization blocks of the CPU 948R for error handling

Organization blocks for reaction to device or program errors [1)]	
Block	**Function and call criterion**
OB 19	Runtime error (LZF): called block not loaded (PB, SB, FB, FX) or attempt to open a data block that is not loaded (DB, DX)
OB 23	Timeout (QVZ) in user program (during direct access to I/O modules)
OB 24	Timeout (QVZ) when updating the process image and transferring interprocessor communication flags.
OB 25	Addressing error (ADF)
OB 26	Cycle time exceeded (ZYK)

Organization blocks for reaction to device or program errors [1]	
Block	**Function and call criterion**
Table 2-4 continued:	
OB 27	Substitution error (SUF)
OB 28	Timeout input byte IB 0 (process interrupts)
OB 29	Timeout distributed I/Os, extended address volume
OB 30	Timeout and parity error (PARE) accessing the user memory
(OB 31)	(set cycle monitoring time) [2]
OB 32	Load and transfer error accessing data blocks (TRAF)
OB 33	Collision of timed interrupts (WEFES/WEFEH)
OB 34	Error setting up a data block (G DB/GX DX)
OB 37	Entry in the error DB

[1] If the OB is not programmed, the CPU changes to the stop mode in the event of an error. EXCEPTION: if OB 19 (logic block not loaded), OB 23, OB 24 (timeout), OB 29 or OB 33 do not exist, there is no reaction!

[2] OB 31 only exists in the CPU 948R for the sake of compatibility.
To set the cycle monitoring time, you should use data block DX0 (refer to Chapter 7)
OB 31 is called once during the start-up, if loaded. You can also use it to set the cycle monitoring time by programming the following STEP 5 operations in it:

```
:L   KF +nnn
:BE
```

nnn is a decimal number. The cycle monitoring time is obtained from "nnn * 100 ms".

Operating system organization blocks of the CPU 948R	
Block	**Function**
OB 100 to OB 119	Internal blocks belonging to operating system

2.2.2 Organization Blocks for Special Functions

The following organization blocks contain special functions of the system program. You **cannot** program these blocks, but simply call them (this applies to all OBs with numbers between 121 and 255!). They do not contain a STEP 5 program. Special function OBs can be called in all logic blocks.

Table 2-5 Overview of the organization blocks of the CPU 948R for special functions

Integrated organization blocks with special functions	
Block:	**Block function:**
OB 121	Set/read time of day (compatible with CPU 946/947)
OB 122	"Disable interrupts" on/off
OB 124	Delete STEP 5 blocks
OB 125	Generate STEP 5 blocks
OB 131	Delete ACCU 1 to ACCU 4
OB 132	Roll up ACCU
OB 133	Roll down ACCU
OB 141	Enable/disable "disable individual timed interrupts"
OB 142	Enable/disable "delay all interrupts"
OB 143	Enable/disable "delay single timed interrupts"
OB 150	Set/read system time (compatible with CPU 928B)
OB 151	Set/read real time interrupt time
OB 153	Set/read time for delayed interrupt
OB 222	Restart cycle monitoring time
OB 254, 255	Copy/duplicate DB and DX data blocks from memory card to user memory

These special functions are described in detail in Chapter 6.

2.3 Function Blocks

Function blocks (FB/FX) are also parts of the user program just like program blocks. FX function blocks have the same structure as FB function blocks and are programmed in the same way.
You use function blocks to implement frequently recurring or very complex functions. In the user program, each function block represents a complex complete function. You can obtain function blocks as follows:

- as a software product from SIEMENS (standard function blocks on diskette - see /11/); with these function blocks you can generate user programs for fast and simple open loop control, signalling, closed loop control and logging;

  or

- you can program function blocks yourself.

Compared with organization, program and sequence blocks, function blocks have the following four essential **differences**:

	OB, PB, SB	**FB/FX**
1.	**Range of operations**	
	only basic operations	- basic operations, - supplementary operations - system operations
2.	**Method of representation**	
	programming and call in STL, LAD, CSF	programming only in AWL
3.	**Name**	
	name environment not possible (only number)	in addition to the number a name with max. 8 chars. can be assigned
4.	**Operands**	
	none	formal operands (block parameters). When the block is called formal operands are assigned actual operands

2.3.1 Structure of Function Blocks

The **block header** (five words) of a function block has the same structure as the headers of the other STEP 5 block types.

The **block body** on the other hand, has a different structure from the bodies of the other block types. The block body contains the function to be executed in the form of a statement list in the STEP 5 programming language. Between the block header and the STEP 5 statements, the function block needs additional memory space for its name and for a list of formal operands. Since this list contains no statements for the CPU, it is skipped with an unconditional jump that the programmer generates automatically. This jump statement is not displayed when the function block is displayed on the PG!

When a function block is called, only the block body is processed.

Absolute or symbolic operands

You can enter operands in a function block in absolute form (e.g. F 2.5) or symbolically (e.g. MOTOR1). You must store the assignment of the symbolic operands in an **assignment list** before you enter the operands in a function block (see /3/).

Fig. 2-4 shows the structure of a function block in the memory of a programmable controller.

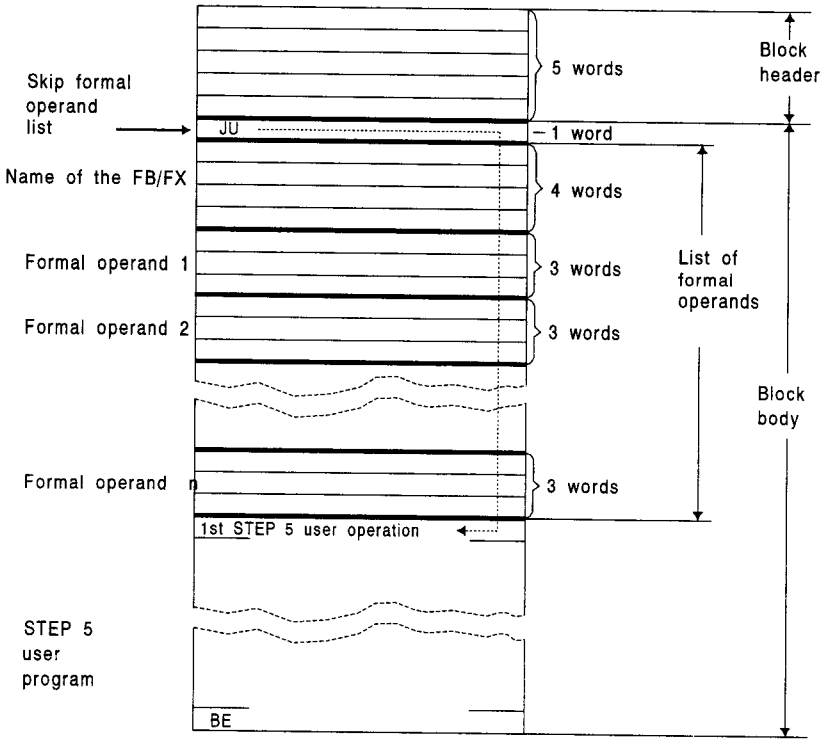


Fig. 2-4 Structure of a function block (FB/FX)

The memory contains all the information that the programmer needs to represent the function block graphically when it is called and to check the operands during parameter assignment and programming of the function block. The programmer rejects incorrect input.

When handling function blocks, distinguish between the following procedures:

- **programming** FB/FX

  and

- **calling** FB/FX and then **assigning actual values** to the parameters.

Distinction: "programming" – "calling and assigning parameters"

When **programming**, you specify the function of the block. You must decide which input operands the function requires and which output results it should transfer to the calling program. You define the input operands and output results as formal operands. These function as tokens.

When a block is **called** by a higher order block (OB, PB, SB, FB, FX), the formal operands (block parameters) are replaced by actual operands; i.e. **parameters** are assigned to the function block.

How to program

IF...	THEN...
You want to program a function block "directly", i.e. without formal operands.	Program it as you would a program or sequence block.
You want to use formal operands in a function block.	Proceed as explained on the following pages. Make sure you keep to the required order: 1. First program the FB/FX with the formal operands and keep it on the PG (offline) or in the CPU memory (online) 2. Then program the block(s) to be called with the actual operands.

2.3.2 Programming Function Blocks

You can program a function block only in the "**statement list**" method of representation. When entering a function block at a programmer, perform the following steps:

Step	Action
1	Enter the **block type** (FB/FX) and the **number** of the function block. Number your function blocks in descending order starting with FB 255, so that they do not collide with the standard function blocks. The standard function blocks are numbered from FB 1 to FB 199.
2	Enter the **name** of the function block. The name can have a maximum of eight characters and must start with a letter.
3	**If the function block is to process formal operands:** Enter the formal operands you require in the block as block parameters. Enter the following information for each formal operand: - the name of the block parameter (maximum 4 characters), - the type of block parameter and the data type of the block parameter (if applicable) You can define a maximum of 40 formal operands.
4	Enter your STEP 5 program in the form of a **statement list** (STL). The formal operands are preceded by an equality sign (e.g. A = X1). They can also be referenced more than once at various positions in the function block.
5	Terminate your program input with the block end operation "BE".

Note
If you change the **order** or the **number** of formal operands in the formal operand list, you must also update all STEP 5 statements in the function block that reference a **formal operand** and also the block parameter list in the calling block!

Program or change function blocks only on diskette or hard disk and then transfer them to your CPU!

Formal operands

The following parameter and data types are permitted as the formal operands of a function block (also known as **block parameters**):

Table 2-5 Permitted formal operands for function blocks

Parameter type	Data type
I = input parameter Q = output parameter	BI/BY/W/D
D = data	KM/KH/KY/KS/KF/ KT/KC/KG
B = block operation T = timer C = counter	none (no type can be specified)

I, D, B, T or **C** are parameters that are indicated to the **left** of the function symbol in graphic representation.
Parameters labelled with **Q** are indicated on the **right** of the function symbol.

The data type indicates whether you are working with bits, bytes, words or double words for I and Q parameters and which data format applies to D parameters (e.g. bit pattern or hexadecimal pattern).

2.3.3 Calling Function Blocks and Assigning Parameters to Them

You can call every function block as often as you want anywhere in your STEP 5 program. You can call function blocks in a statement list or in one of the graphic methods of representation (CSF or LAD).

To call a function block and assign parameters to it, perform the following steps:

Step	Action	Reaction on PG
1	Make sure that the called function block exists either in the PG memory (offline) or in the CPU memory (online).	none
2	Enter the call statement for the function block in the block where the call is to originate. You can program a function block call in an organization, program or sequence block or in another function block.	After you enter the call statement (e.g. JU FB200), the name of the relevant function block and the formal operand list appear automatically.
3	Assign the **actual** operand relevant to this call to each of the formal operands, i.e. you assign **parameters** to the function block. These actual operands can be different for separate calls (e.g. inputs and outputs for the first call of FB 200, flags for the second call). Using the formal operand list, you assign the required actual operands for each function block call.	none

Unconditional/conditional call

Unconditional call	Conditional call
"**JU FBn**" for FB function blocks or "**DOU FXn**" for FX extended function blocks: the referenced function block is processed regardless of the previous result of logic operation (RLO).	"JC FBn" for FB function blocks or "**DOC FXn**" for FX extended function blocks: the referenced function block is only processed when the result of logic operation RLO = 1. If RLO = 0 the block call is not executed. Regardless of whether the block call is executed or not, the RLO is always set to "1".
After the unconditional or conditional call, the RLO can no longer be combined logically. However, it is carried over to the called function block with the jump and can be evaluated there.	

Permitted actual operands

Which operands can be assigned as **actual operands** is shown in the following table

.

Table 2-6 Permitted actual operands for function blocks

Parameter type	**Data type**		**Actual operands permitted**		
I, Q	BI	for an operand with bit address	I	n.m	input
			Q	n.m	output
			F	n.m	flag
	BY	for an operand with byte address	IB	n	input byte
			QB	n	output byte
			FY	n	flag byte
			DL	n	data byte left
			DR	n	data byte right
			PY	n	peripheral byte
			OY	n	byte from extended periphery
	W	for an operand with word address	IW	n	input word
			QW	n	output word
			FW	n	flag word
			DW	n	data word
			PW	n	peripheral word
	D	for an operand with double word address	OW	n	word from extended periphery
			ID	n	input double word
			QD	n	output double word
			FD	n	flag double word
			DD	n	data double word
D	KM	for a binary pattern (16 bits)	Constants		
	KY	for two absolute numbers, one byte each, each in the range from 0 to 255			
	KH	for a hexadecimal pattern with a maximum of four digits			
	KS	for two alphanumeric characters			
	KT	for timer value (BCD-coded) units .0 to .3 and values 0 to 999			
	KC	for a counter value 0 to 999			

Parameter type	Data type	Actual operands permitted
Table 2-6 continued:		
D (Cont.)	KF for a fixed point number -32768 to +32767 KG for a floating point number[1)]	Constants
B	Data type designation not possible	DB n Data block; the operation C DB n is executed FB n Function block (permitted only without parameters) called unconditionally (JU . .n) OB n Organization block called unconditionally (JU . .n) PB n Program blocks - called unconditionally (JU . .n) SB n Sequence blocks - called unconditionally (JU . .n)
T	Data type designation not possible	T 0 to 255 Timer
C	Data type designation not possible	C 0 to 255 Counter

1) $\pm 0.1469368 \times 10^{-38}$ to $\pm 0.1701412 \times 10^{39}$

Note
S flags are **not** permitted as actual operands for function blocks.

After the jump to a function block, the actual operands from the block then called are used in the function block program instead of the formal operands.
This feature of programmable function blocks allow them to be used for a wide variety of purposes in your user program.

Examples

Example 1: the following (complete) example is intended to further clarify the programming and calling of a function block and the assignment of parameters to it. You yourself can easily try out the example.

Programming the function block FB 202:

```
FB 202

SEGMENT 1
NAME EXAMPLE
DECL : INP1   I/Q/D/B/T/C: I    BI/BY/W/D: BI
DECL : INP2   I/Q/D/B/T/C: I    BI/BY/W/D: BI
DECL : OUT1   I/Q/D/B/T/C: Q    BI/BY/W/D: BI

     :A= INP1
     :A= INP2
     :== OUT1

     :
     : BE
```

Formal operand list

STEP 5 state-ments

Formal operands

Parameter type

Data type

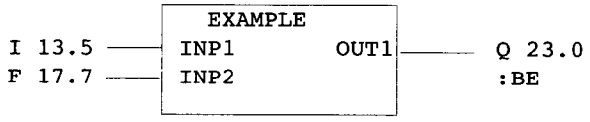
Function block FB 202 is called and has parameters assigned to it in program block PB 25:

STL method of representation

```
PB 25
SEGMENT 1

     : JU FB 202
NAME : EXAMPLE
INP1 : I 13.5
INP2 : F 17.7
OUT1 : Q 23.0
     : BE
```

CSF/LAD method of representation



Formal operands

Actual operands

The following operations are executed after the jump to FB 202

```
: A  I 13.5
: A  F 17.7
: =  Q 23.0
:BE
```

Example 2: calling a function block and assigning parameters to it with the STL and CSF/LAD methods of representation in a program block.

STL method of representation

```
PB 25
SEGMENT 1
      :
      :  C  DB 5
      :
      :  JU    FB 201
NAME  :  REQUEST
DATA  :        DW   1
RST   :        I    3.5
SET   :        F    2.5
MTIM  :        T    2
TIME  :        KT   010.1
TRAN  :        DW   2
BEC   :        Q    2.3
LOOP  :        Q    6.0
      :  BE
```

Formal operands

Actual operands

CSF/LAD method of representation

```
PB 25
SEGMENT 1
                  FB 201
                 REQUEST
DW 1        DATA          TRAN        DW  2
I  3.5      RST           BEC         Q   2.3
F  2.5      SET           LOOP        Q   6.0
T  2        MTIM                      :BE
KT 010.1    TIME
```

2.3.4 Special Function Blocks

Apart from the function blocks that you program yourself, you can order **standard function blocks** as a finished software product. These contain standard functions for general use (e.g. signalling functions and sequence control).
Standard function blocks are assigned numbers FB 1 to FB 199.

If you order standard function blocks, remember the special instructions in the accompanying description (i.e. areas assigned and conventions etc.).

The standard function blocks for the S5-155H are listed in catalog ST 57 /11/.

Example

Floating point root extractor RAD:GP FB 6

The function block RAD:GP extracts the root of a floating point number (8-bit exponent and 24-bit mantissa). It forms the square root. The result is also a floating point number (8-bit exponent and 24-bit mantissa). The least significant bit of the mantissa is not rounded up or down.

If applicable, for the rest of the processing, the function block sets the "radicand negative" identifier.

Numerical range:

Radicand - 0.1469368 Exp. -38 to +0.1701412 Exp. +39

Root +0.3833434 Exp. -19 to +0.1304384 Exp. +20

Function: $Y = \sqrt{A}$

Y = SQRT; A = RADI

Calling the function block FB 6:

In the example, the root is extracted from a floating point number that is located in DD5 of DB 17 with an 8-bit exponent and a 24-bit mantissa. The result, another 32-bit floating point number, is written to DD 10. Prior to this, the appropriate data block must be opened. The parameter VZ (parameter type: Q, data type: BI) indicates the sign of the radicand: VZ = 1 for a negative radicand.

Occupied flag words: FW 238 to FW 254.

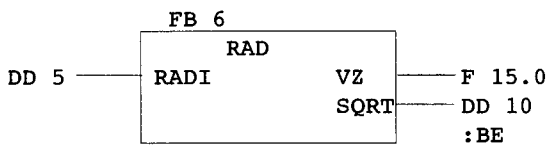
Continued on next page

"Floating point root extractor" continued:

STL method of representation

```
Seg-    :  C DB 17
ment    :
1       :***
        :  JU FB 6
Seg-    NAME : RAD : GP
ment    RADI : DD 5
2       VZ   : F 15.0
 *)     SQRT : DD 10
```

LAD method of representation



DD= data double word

*) Must be located in separate segments, since the operation "C DB 17" in segment 1 cannot be converted to LAD/CSF.

2.4 Data Blocks

Data blocks (DB) or extended data blocks (DX) are used to store the fixed or variable data with which the user program works. **No** STEP 5 operations are processed in data blocks.

The data of a data block includes the following:

- various bit patterns (e.g. for status of a controlled process)
- numbers (hexadecimal, binary, decimal) for timer values or arithmetic results
- alphanumeric characters, e.g. for message texts.

Structure of a data block

A data block (DB/DX) consists of the following parts:

- block preheader (DV, DXV),
- block header
- block body.

Block preheader

The **block preheader** is created automatically on the hard or floppy disk of the PG and not transferred to the CPU. It contains the data formats of the data words entered in the block body. You have no influence over the creation of the block preheader.

> **Note**
> When you transfer a data block from the PLC to diskette or hard disk, the corresponding block preheader can be deleted. For this reason, you must never modify a data block with different data formats in the PLC and then transfer it back to diskette, otherwise all the data words in the DB are automatically assigned the data format you selected in the presets screen form.

Block header

The **block header** occupies five words in the memory and contains the following:

- the block identifier
- the programmer identifier
- the block type and the block number
- the library number
- the block length (including the length of the block header).

Block body

The **block body** contains the data words with which the user program works. These data words are in ascending order in the block body, starting with data word DW 0. Each data word occupies one word (16 bits) in the memory.

Maximum length

A data block can occupy a total of maximum 32 767 words (including header) in the CPU memory. When you use your programmer to enter and transfer data blocks, remember the size of your CPU memory!

2.4.1 Creating Data Blocks

To create a data block, perform the following steps:

Step	Action
1	Enter the block type (DB/DX) and data block **number** (2 or 3 to 255).
2	Enter individual **data words** in the data format you require. (Do **not** complete your input of the data words with a BE statement!)

Note
Data blocks DB 0, DB 1, DX 0, DX 1 and DX 2 are reserved for specific functions and you cannot use them freely for other functions (see Section 2.4.3)!

Table 2-7 Data formats permitted in a data block

Type	Data format	Examples
KM	Bit pattern	00100110 00111111
KH	Hexadecimal	263F
KY	2 Bytes	038,063
KF	Fixed point number	+09791
KG	Floating point number	+1356123+12
KS	Character	?!ABCD123-+.,%
KT	Timer value	055.2
KC	Counter value	234

2.4.2 Opening Data Blocks

You can only open a data block (DB/DX) **unconditionally**. This is possible within an organization, program, sequence or function block. You can open a specific data block more than once in a program.

To open a data block, perform the following steps:

IF...	THEN...
You want to open a **DB** data block	Type in the STEP 5 operation **"C DB.."**
You want to open a **DX** data block	Type in the STEP 5 operation **"CX DX.."**

Validity of a data block

After you open a data block, all statements that follow with the operand area **'D'** refer to the opened data block.

The opened data block also remains valid when the program is continued in a different block following a block call.

If a second data block is opened in this new block, the second data block is **only** valid in the newly called block from the point at which it is called. After program execution returns to the calling block, the old data block is once again valid.

Access

You can **access** the data stored in the opened data block during program execution using **binary logic operations, set/reset operations, load or transfer operations** (refer to Chapter 3 for more detailed information).

With a **binary operation**, the addressed data word bit is used to form the RLO. The content of the data word is not changed.

With a **set/reset operation**, the addressed data word bit is assigned the value of the RLO. The content of the data word may be changed.

A **load operation** transfers the contents of the referenced data word into ACCU 1. The contents of a data word are not changed.

A transfer operation transfers data from ACCU 1 to the referenced data word. The old contents of the data word are overwritten.

Note
Before accessing a data word, you must open the data block you require in your program. This is the only way that the CPU can find the correct data word.
The referenced data word must be contained in the opened block, otherwise the system program detects a load or transfer error.

With load and transfer operations, you can only access data word numbers up to 255!

An opened data block remains valid until one of the following events occur:

a) a second data block is opened

or

b) the block, in which the data block was opened, is completed with 'BE', 'BEC' or 'BEU'.

Examples

Example 1: transferring data words

You want to transfer the contents of data word DW 1 from data block DB 10 to data word DW 1 of data block DB 20.

Enter the following statements:

```
:C   DB 10   (open DB 10)
:L   DW 1    (load the contents of DW 1 into
:            ACCU 1)
:C   DB 20   (open DB 20)
:T   DW 1    (transfer the contents of ACCU 1 to
:            DW 1)
:
```

Example 2: range of validity of data blocks
(Fig. 2-5)

Data block DB 10 is opened in program block PB 7 (C DB 10). During the subsequent program execution, the data of this data block are processed.

After the call (JU PB 20) program block PB 20 is processed. Data block DB 10, however, remains valid. The data area only changes when data block DB 11 (C DB 11) is opened.
Data block DB 11 now remains valid until the end of program block PB 20 (BE).

After the jump back to program block PB 7, data block DB 10 is once again valid.

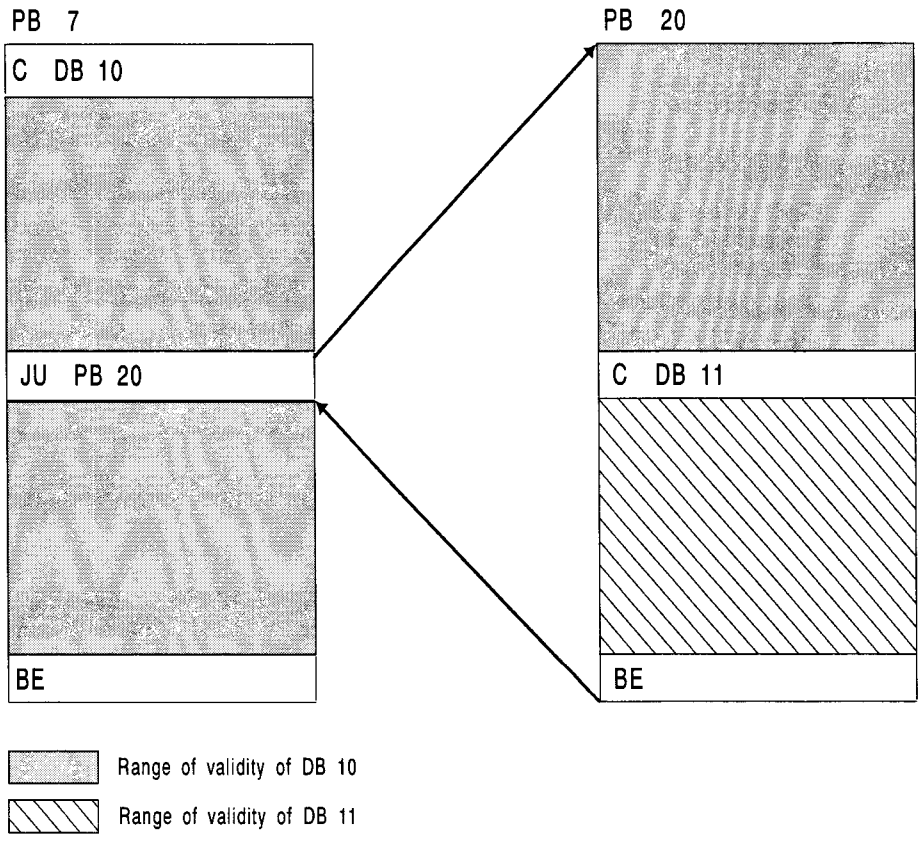


Fig. 2-5 Range of validity of an opened data block

2.4.3 Special Data Blocks

On the CPU 948 data blocks DB 0, DB 1, DX 0, DX 1 and DX 2 are reserved for special functions. They are managed by the system program and you cannot use them freely for other functions.

DB 0

- **Data block DB 0** (see Section 8.3.2)

  Data block DB 0 contains the address list with the start addresses of all blocks that are located in the data block RAM of the CPU. The system program generates this address list during initialization (following each POWER UP or OVERALL RESET) and it is updated automatically when you use a programmer to change data blocks or generate a new data block.

DB 1

- **Data block DB 1** (see Section 10.1.6)

  Data block DB 1 contains the list of digital inputs/outputs (P peripheral with relative byte addresses from 0 to 127) and the interprocessor communication (IPC) flag inputs and outputs that are assigned to the CPU. If applicable, the block may also contain a timer field length.

  DB 1 **can** have parameters assigned and be loaded as follows:
  to reduce the cycle time in single processor operation, since only the inputs, outputs or timers entered in DB1 are updated.

  DB 1 must be assigned parameters and loaded as follows:
  a) for multiprocessing
  b) when IPC flags exist with CPs

DX 0

- **Data block DX 0** (see Chapter 7)

  If you assign parameters to data block DX 0 and load it, you can change the defaults of certain system program functions (e.g. the start-up procedure) and adapt the performance of the system program to your particular application.

DX 1

- **Data block DX 1**

  Parameter assignment data block for COM 155H (see Part I of Volume 1)

DX 2

- **Data block DX 2**

  Reserved.

Program Execution 3

Contents of Chapter 3

3.1 Principle of Program Execution ... 3 - 4

3.2 Program Organization ... 3 - 5

3.3 Storing Program and Data Blocks ... 3 - 10

3.4 Processing the User Program ... 3 - 11

3.4.1 Definition of Terms used in Program Execution ... 3 - 12

3.5 STEP 5 Operations with Examples ... 3 - 15

3.5.1 Basic Operations ... 3 - 19
- Binary logic operations ... 3 - 19
- Set/reset operations ... 3 - 20
- Load and transfer operations ... 3 - 21
- Timer and counter operations ... 3 - 26
- Arithmetic operations ... 3 - 31
- Comparison operations ... 3 - 32
- Block operations ... 3 - 32
- NOP/display/stop operations ... 3 - 34

3.5.2 Programming Examples in the STL, LAD and CSF Methods of Representation ... 3 - 35

3.5.3 Supplementary Operations ... 3 - 51
- Binary logic operations ... 3 - 52
- Digital logic operations ... 3 - 52
- Bit test operations ... 3 - 53
- Set/reset operations ... 3 - 54
- Timer and counter operations ... 3 - 55
- Load and transfer operations ... 3 - 57
- Arithmetic operations ... 3 - 59

3.5.4 Executive Operations ..... 3 - 61
Jump operations ..... 3 - 61
Shift operations ..... 3 - 63
Conversion operations ..... 3 - 65
Decrement/increment ..... 3 - 68
Processing operations ..... 3 - 68
I/O operations ..... 3 - 74
Other operations ..... 3 - 74

Program Execution

3

This chapter is intended for readers who do not yet have any great experience in using the programming language. The chapter therefore deals with the basics of STEP 5 programming and explains in detail (with examples) the STEP 5 operations for the CPU 948R.

Experienced readers who require more information about a specific STEP 5 operation listed in the Pocket Guide can refer to the reference section in 3.5.

3.1 Principle of Program Execution

You can process your STEP 5 user program in various ways.

Cyclic program execution is most common with programmable controllers (PLCs). The system program runs through a program loop (the cycle, refer to Section 3.4) and calls organization block OB 1 cyclically in each loop (refer to Fig. 3-1).

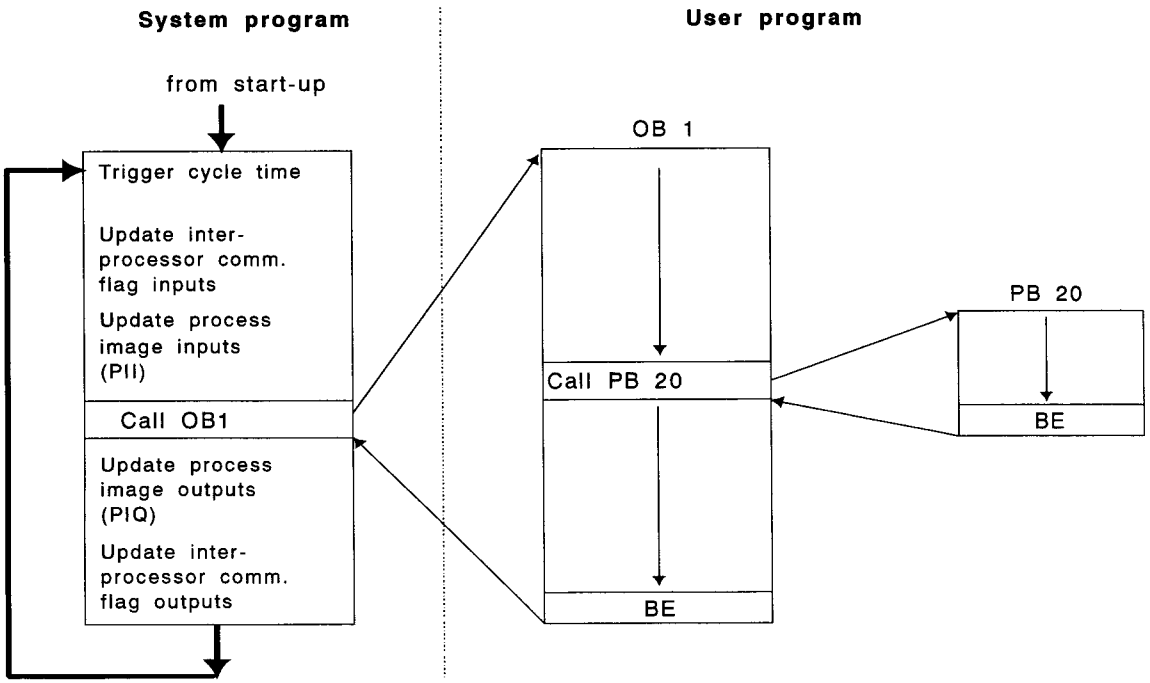


Fig. 3-1 Principle of cyclic program execution

3.2 Program Organization

Program organization allows you to specify which conditions affect the processing of your blocks and the order in which they are processed. Organize your program by programming organization blocks with conditional or unconditional calls for the blocks you require.

You can call additional program, function and sequence blocks in any combination in the program of individual organization, program, function and sequence blocks. You can call these one after another or nested in one another.

For maximum efficiency, you should organize your program to emphasise the most important program structures and in such a way that you can clearly recognize parts of the controlled system which are related in the software.

Figs. 3-2 and 3-3 are examples of a program structure.

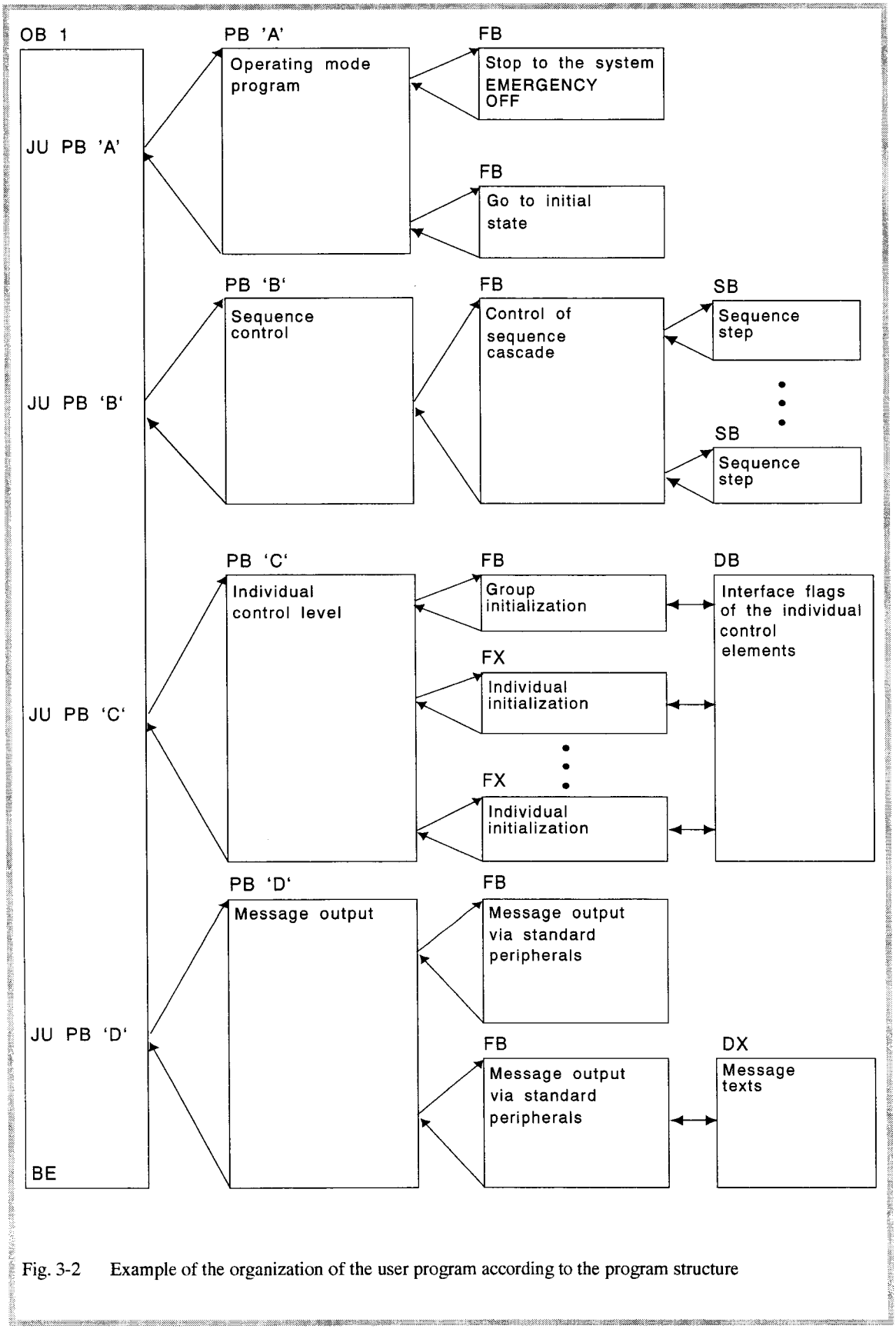


Fig. 3-2 Example of the organization of the user program according to the program structure

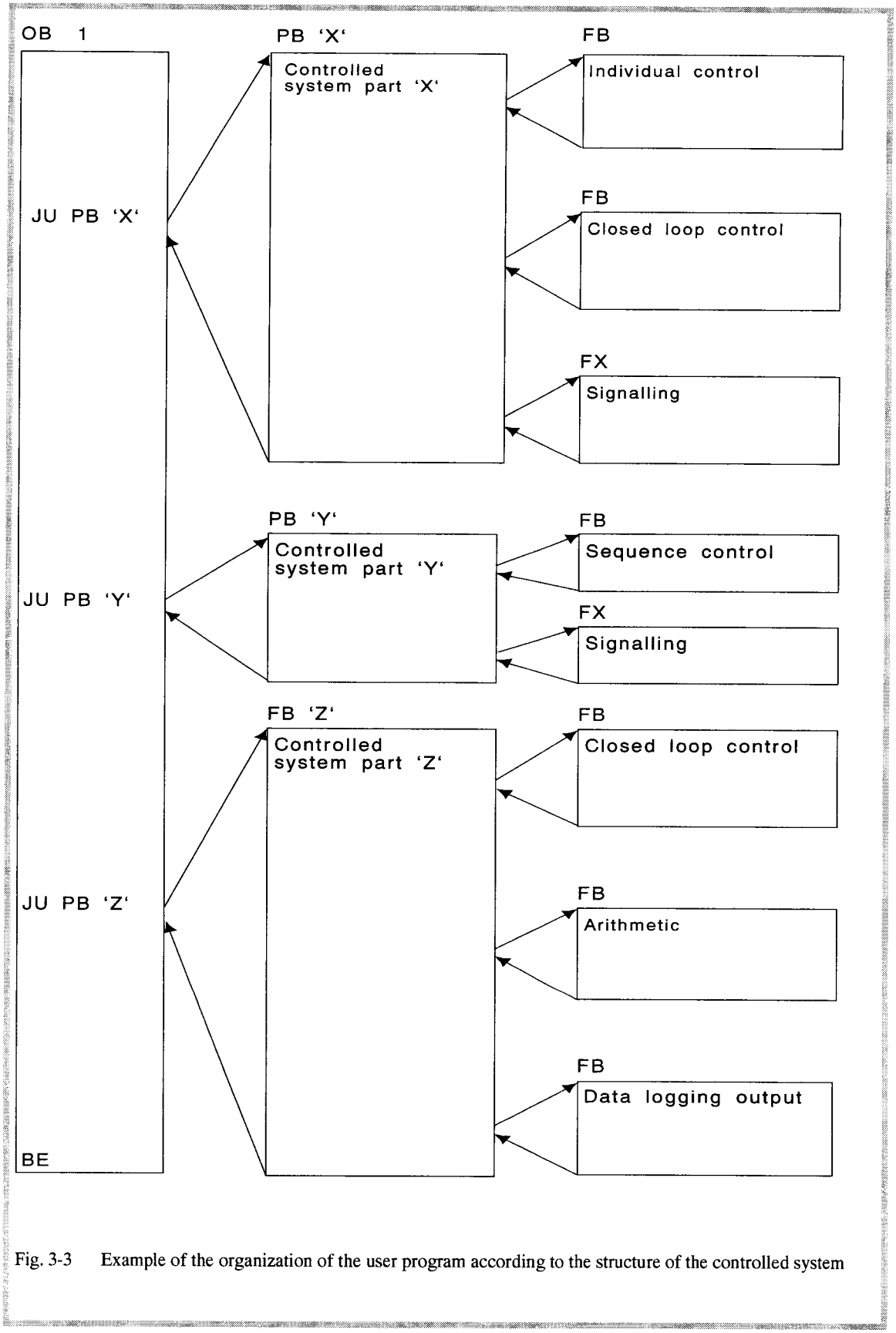
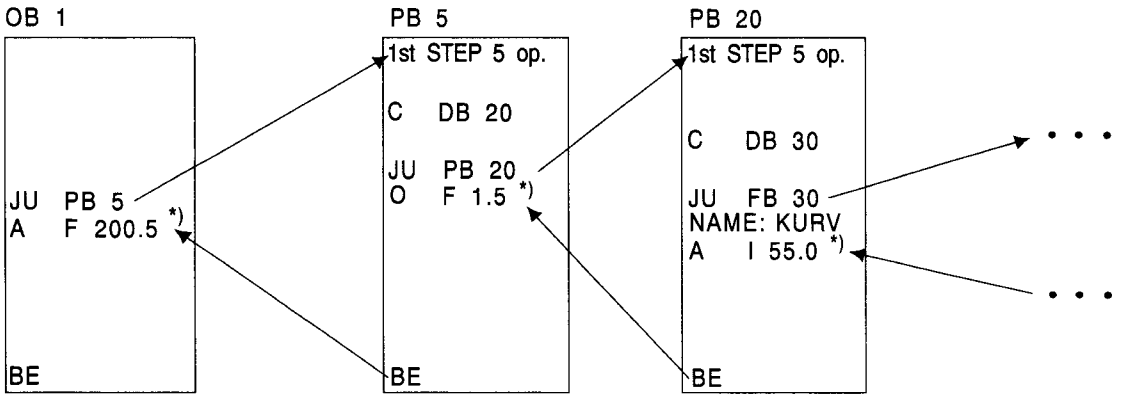


Fig. 3-3 Example of the organization of the user program according to the structure of the controlled system

Nesting blocks

Fig. 3-4 shows the principle of nested block calls.



*) Operation to which the program returns

Fig. 3-4 Nested logic block calls

Block addresses

A block start address specifies the location of a block in the user memory. For logic blocks, this is the address of the memory location containing the first STEP 5 operation (with FB and FX, the JU operation via the formal operand list); with data blocks, it is the address of the first data word.

To enable the CPU to locate the called block in the memory, the start addresses of all valid blocks are entered in the block address list in data block DB 0. DB 0 is managed by the system program, you cannot call it yourself.

The CPU stores a **return address** every time a new block is called. After the new block has been processed, this return address enables the program to find the block from which the call originated. The return address is the address of the memory location containing the next STEP 5 statement after the block call. The CPU also stores the **start address and length of the data block** valid at this location.

Nesting depth

You can only nest 60 blocks within one another. If more than 60 blocks are called, the CPU signals an error and goes to the stop mode.

Example of nesting depth

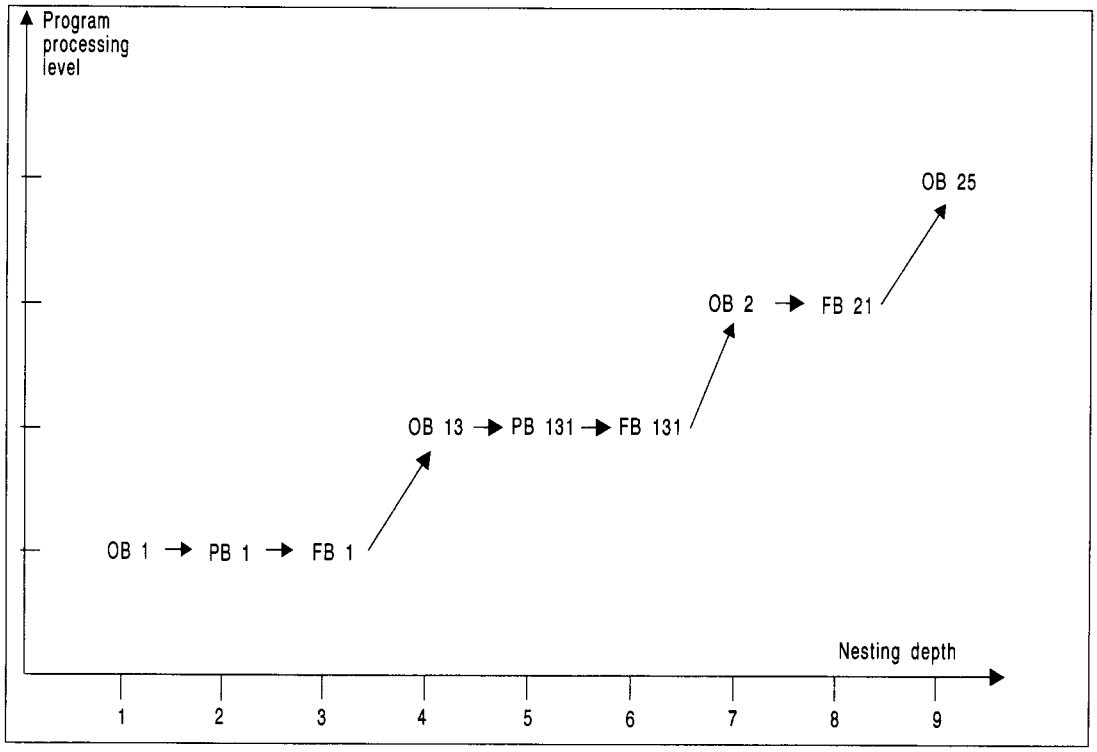


Fig. 3-5 Example of block nesting depth

You can determine the nesting depth of your program as follows:

- Add all the organization blocks you have programmed (in the example: 4 OBs).
- Add the nesting depth of the individual organization blocks (in the example: 2 + 2 + 1 + 0 = 5).
- Add the two amounts together to obtain the program nesting depth (in the example: 4 + 5 = nesting depth 9). It must not exceed a value of 40.

3.3 Storing Program and Data Blocks

On the CPU 948R, the user program runs solely in the internal RAM. The user program including data blocks must, therefore, be loaded in the CPU 948R user memory.

How do I load programs and data blocks in the internal RAM?

You can use the following methods:

- You can load the individual logic and data blocks in the RAM using your PG.
- You can program a memory card (flash EPROM!) with your complete program including data blocks on the PG and then insert the card in the receptacle on the CPU.
  If you do an overall reset on the CPU (refer to Chapter 4) the complete contents of the memory card are loaded "1:1" in the internal RAM.
- You loaded your program in the internal RAM with the PG or from the memory card with an OVERALL RESET. You can then load additional blocks with the PG or replace existing blocks.

Note
You can only program the memory card on the PG. Use the PG software from version 6 upwards. When programming, the PG must be in the mode "WORD FIELD" (refer to the STEP 5 manual /3/).



Caution
If you have changed or added blocks using the PG after loading your program from the memory card, these changes are reversed by the next **OVERALL RESET**, since the memory is **overwritten** again with the contents of the memory card.

3.4 Processing the User Program

The complete software on the CPU (consisting of the system program and the STEP 5 user program) has the following tasks:

- CPU START-UP
- Controlling an automation process by continuously repeating operations (CYCLE).
- Controlling an automation process by reacting to events occurring sporadically or at certain times (interrupts) and reacting to errors.

For all three tasks, you can select special parts of your program to run on the CPU by programming user interfaces (organization blocks OB 1 to OB 35 - refer to Section 2.2.3).

START-UP

Before the CPU can start cyclic program execution, an initialization must be performed to establish a defined initial status for cyclic program execution and, for example, to specify a time base for the execution of certain functions. The way in which this initialization is performed depends on the event that led to a START-UP and on settings that you can make on your CPU. For more detailed information, refer to Chapter 4.

You can influence the START-UP procedure of your CPU by programming organization blocks OB 20, OB 21 and OB 22 or by assigning parameters in DX 0 (refer to Chapter 7).

CYCLE

Following the START-UP, the system program goes over to cyclic processing. It is responsible for background functions required for the automation tasks (refer to Fig. 3-1 at the beginning of this section). After the system functions have been executed at the beginning of a CYCLE, the system program calls organization block OB 1 or function block FB 0 as the cyclic user program. You program the STEP 5 operations for cyclic processing in this block.

Reactions to interrupts and errors

To allow you to specify the reactions to interrupts or errors, special organization blocks (OB 2 to OB 18 for interrupt servicing, OB 19 and OB 23 to OB 34 for reactions to errors) are available on the CPU 948R. You can store an appropriate STEP 5 program in these blocks.

When interrupts or errors are to be processed, the system program activates the corresponding organization block during cyclic processing. This means that the cyclic processing is interrupted to service an interrupt or to react to an error. The nesting of the organization blocks has a fixed priority (for further information, refer to Chapters 4 and 5).

In addition to the organization blocks, you can also influence the reaction of the CPU to interrupt servicing by assigning parameters in data block DX 0.

Organization blocks OB 1 to OB 39 can be called by the system program as soon as they are loaded in the program memory (**also during operation**).
If the OBs are not loaded, there is either no reaction from the CPU or (in the event of errors) it goes to the stop mode (refer also to Section 5.4).

You can also load data block DX 0 into the program memory during operation like the organization blocks. **It is, however, only effective after the next COLD RESTART.** If DX 0 is not loaded, the standard settings apply (refer to Chapter 7).

3.4.1 Definition of Terms used in Program Execution

Cycle time

The cycle begins when the cycle monitoring time is triggered and ends with the next trigger. The time that the CPU requires to execute the program between two triggers is called the cycle time. The cycle time consists of the runtime of the system program and the runtime of the user program.

The cycle time therefore includes the following:

- the time required to process the cyclic program (system and user program),
- the time required to process interrupts (e.g. real time interrupt),
- the time required to process interruptions (errors).

Cycle time monitoring

The CPU monitors the cycle time in case it exceeds a maximum value. The standard setting for this maximum value is 200 ms. You can set the cycle time monitoring yourself or restart it during user program execution (refer to DX 0/Chapter 7 and special function OB OB 222/Section 6.16).

Process input and output image (PII and PIQ)

The process image of the inputs and outputs is a memory area in the internal RAM.
Before cyclic execution of the user program begins, the system program reads the signal states of the input peripheral modules and transfers them to the process input image. The user program evaluates the signal states in the process input image and then sets the appropriate signal states for the outputs in the process output image. After the user program has been processed, the system program transfers the signal states of the process output image to the output peripheral modules.

Buffering the I/O signals in the process image of the inputs and outputs avoids a change in a bit within a program cycle from causing the corresponding output to "flutter".

The process image is therefore a memory area whose contents are output to the peripherals and read in from the peripherals **once per cycle.**

Note
The process image only exists for input and output bytes of the "P" peripherals with byte addresses from 0 to 127!

Interprocessor communication (IPC) flags

IPC flags exchange data between the CPU and some communications processors.

The system program reads the input IPC flags of the CPU before cyclic execution of the user program begins. After the STEP 5 program is processed, the system program transfers the output IPC flags to the communications processors.

You define the input and output IPC flags when you create data block DB 1 (STEP 5 screen on the PG).

Interrupt events

Cyclic program execution can be interrupted by the following:

- time-controlled program execution (delayed interrupt, cyclic timed interrupts, real time interrupts),
- interrupt-driven program execution (process interrupt, system interrupt).

The cyclic program can be interrupted or even aborted completely by the following:

- a program error
- operator intervention (using the PG function or STOP selector),
- a stop operation

3.5 STEP 5 Operations with Examples

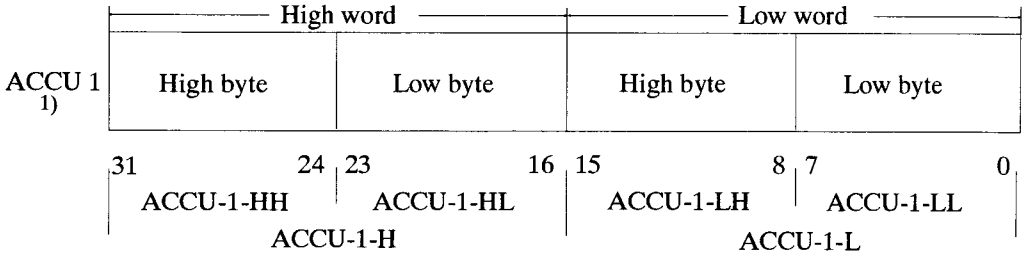
A STEP 5 operation consists of the operation and an operand. The operation specifies **what** the CPU is to do (operation). The operand specifies **with what** an operation is to be executed.

STEP 5 operations can be divided into the following groups:

- **basic operations** (can be used in **all** logic blocks),
- **supplementary operations,**
- **executive operations** (can only be used in FB/FX function blocks),
- semaphore operations (can only be used in FB/FX function blocks).

Accumulators as working registers

The CPU 948R has four accumulators, ACCU 1 to ACCU 4. Most STEP 5 operations use two 32-bit registers (ACCU 1 and ACCU 2) as the source of operands and the destination for results.



The STEP 5 operation to be carried out affects the accumulators, e.g.:

- ACCU 1 is always the destination in load operations. A load operation shifts the old contents of ACCU 1 to ACCU 2 (stack lift). Accumulators 3 and 4 are not changed by any load operations.

[1] analogous for ACCU 2 to ACCU 4

- Arithmetic operations combine the contents of ACCU 1 with those of ACCU 2, write the result to ACCU 1 and transfer the contents of ACCU 3 to ACCU 2 and the contents of ACCU 4 to ACCU 3 (stack drop). In 16-bit fixed point arithmetic, only the low word or ACCU 3 is transferred to the low word of ACCU 2 and the low word of ACCU 4 to the low word of ACCU 3.

- When a constant is added (ADD BF/KF/DH) to the contents of ACCU 1, the accumulators 2, 3 and 4 are not changed.

Condition codes

STEP 5 operations either set or evaluate condition codes. The condition codes are written to a condition code byte. Two groups of condition codes can be distinguished: condition codes of digital operations (word condition codes - bits 4 to 7 in the condition code byte) and condition codes from binary and executive operations (bit condition codes - bits 0 to 3 in the condition code byte). You can see how the various condition codes are influenced or evaluated by STEP 5 operations be referring to the operation list /1/.

You can display the condition code byte on a programmer using the "STATUS" online function (refer to Section 11.2.3). The byte has the following structure:

Word condition codes				**Bit condition codes**			
CC 1	CC 0	OV	OS	OR	STA	RLO	$\overline{\text{ERAB}}$
Bit 7	6	5	4	3	2	1	0

Bit condition codes

- **$\overline{\text{ERAB}}$** First bit scan

  A logic operation sequence containing binary operations always **begins** with the **first bit scan**, following which a **new** RLO is formed. The bit condition code $\overline{\text{ERAB}}$ = 1 is then set. While the remaining logic operations in the sequence are being performed, $\overline{\text{ERAB}}$ remains set to 1 and the RLO cannot be changed by these logic operations.

  The active sequence of logic operations is **terminated** by a binary set/reset operation (e.g. S Q 5.0). The set/reset operation sets $\overline{\text{ERAB}}$ to 0; the RLO can be evaluated (e.g. by RLO-dependent operations) but can no longer be combined logically. The next binary logic operation following a binary set/reset operation is once again a first bit scan.

Example of $\overline{ERAB}$

```
:A    I  1.0     ERAB is set to '1',
:                the new RLO is formed by
:                an AND operation
:O    I  6.3     The RLO is influenced by
:                an OR operation
:AN   I  2.1     The RLO is influenced by
:                an AND NOT operation.
:S    Q  2.4     ERAB is set to '0',
:                the sequence is now complete
:JC   FB 150     The function block is called
:                dependent on the RLO.
:
:
```


Other bit condition codes

- **RLO** Result of logic operation

  This is the result of bit logic operations. It is the truth statement for comparison operations (refer to operations list, binary logic operations or comparison operations).

- **STA** Status

  For bit operations, this indicates the logical status of the bit just scanned or set. The status is updated in binary logic operations - except for A(, O(,), O and for set/reset operations.

- **OR** Or

  Internal CPU bit for handling "AND before OR" logic operations.

Word condition codes

- **OV** Overflow

  This indicates whether the permissible number range was exceeded during the arithmetic operation just completed.

- **OS** Stored overflow

  The overflow bit is stored. It can be used in several arithmetic operations to indicate whether an overflow occurred at any point during the operations.

- **CC 1 and CC 0**

  These are the result condition codes that you can interpret from the following table:

> **Note**
> To evaluate the condition codes directly, comparison and jump operations are available (refer to Sections 3.5.1 and 3.5.3).

Table 3-1 Result condition codes of STEP 5 operations

Word condition codes		Arith-metical operations	Digital logic operations	Com-parison operations	Shift operations	For SED, SEE	Jump operations executed
CC 1	CC 0						
0	0	Result = 0	Result = 0	ACCU 2 = ACCU 1	Shifted bit = 0	Semaphore is set	JZ
0	1	Result < 0	–	ACCU 2 < ACCU 1	–	–	JM JN
1	0	Result > 0	Result ≠ 0	ACCU 2 > ACCU 1	Shifted bit = 1	Semaphore is set or enabled	JP JN
1	1	Division by 0	–	–	–	–	–

> **Note**
> When a change of level takes place, e.g. servicing a timed interrupt, all accumulators and the bit and word condition codes (RLO etc.) are saved and loaded again when the interrupted level is resumed.

3.5.1 Basic Operations

You can use the basic operations in **all** logic blocks and all methods of representation (STL, LAD, CSF).

Binary logic operations

Table 3-2 Binary logic operations

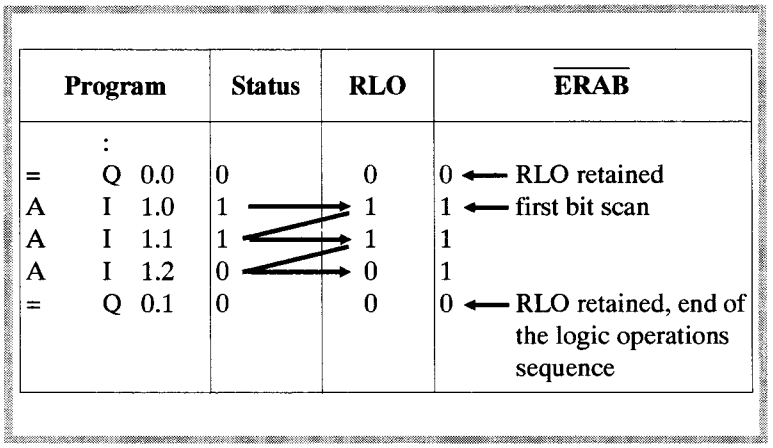
Operation	Operand	Function
A		AND logic operation after scanning for signal state "1"
O		OR logic operation after scanning for signal state "1"
	I 0.0 to 127.7	of an input in the PII
	Q 0.0 to 127.7	of an output in the PIQ
	F 0.0 to 255.7	of a flag bit
	S 0.0 to 4095.7	of an S flag bit
	D 0.0 to 255.15	of a data word bit
	T 0 to 255	of a timer
	C 0 to 255	of a counter
AN		AND logic operation after scanning for signal state "0"
ON		OR logic operation after scanning for signal state "0"
	I 0.0 to 127.7	of an input in the PII
	Q 0.0 to 127.7	of an output in the PIQ
	F 0.0 to 255.7	of a flag bit
	S 0.0 to 4095.7	of an S flag bit
	D 0.0 to 255.15	of a data word bit
	T 0 to 255	of a timer
	C 0 to 255	of a counter
O	–	Combine AND operations through logic OR
O A(O( )	–	ANDing of expressions in parentheses ORing of expressions in parentheses Close parenthesis (to complete the bracketed expression) Maximum of 8 levels are permitted, i.e. 7 opened brackets

RLO formation

The binary logic operations generate the result of logic operation (RLO).
At the beginning of a logic sequence, the RLO only depends on the signal state scanned (first scan) and not on the type of logic operation (O = OR, A = AND).

Within a sequence of logic operations, the RLO is formed from the type of operation, previous RLO and the scanned signal state. A sequence of logic operations is completed by an operation (e.g. set/reset operations) which retains the RLO (ERAB = 0). Following this, the RLO can be evaluated but cannot be further combined.

Example



Set/reset operations

Table 3-3 Set/reset operations

Operation	Operand	Function
S		Set if RLO = 1
R		Reset if RLO = 1
	I 0.0 to 127.7	an input in the PII
	Q 0.0 to 127.7	an output in the PIQ
	F 0.0 to 255.7	a flag
	S 0.0 to 4095.7	an S flag
	D 0.0 to 255.15	a bit in the data word
=		The RLO is assigned to
	I 0.0 to 127.7	an input in the PII
	Q 0.0 to 127.7	an output in the PIQ
	F 0.0 to 255.7	a flag
	S 0.0 to 4095.7	an S flag
	D 0.0 to 255.15	a bit in the data word

Load and transfer operations

Table 3-4 Load and transfer operations/part 1

Operation	Operand		Function
L T			Load Transfer
	IB	0 to 127	an input byte from/to the PII
	IW	0 to 126	an input word from/to the PII
	ID	0 to 124	an input double word from/to the PII
	QB	0 to 127	an output byte from/to the PIQ
	QW	0 to 126	an output word from/to the PIQ
	QD	0 to 124	an output double word from/to the PIQ
	FY	0 to 255	a flag byte
	FW	0 to 254	a flag word
	FD	0 to 252	a flag double word
	SY	0 to 4095	an S flag byte
	SW	0 to 4094	an S flag word
	SD	0 to 4092	an S flag double word
	DR	0 to 255	the right byte of a data word from/to DB,DX
	DL	0 to 255	the left byte of a data word from/to DB,DX
	DW	0 to 255	a data word from/to DB, DX
	DD	0 to 254	a data double word from/to DB, DX
	PY	0 to 127	a peripheral byte of the digital inputs/outputs (P area)
	PY	128 to 255	a peripheral byte of the analog or digital inputs/outputs (P area)
	PW	0 to 126	a peripheral word of the digital inputs/outputs (P area)
	PW	128 to 254	a peripheral word of the analog or digital inputs/outputs (P area)
	OY	0 to 255	a byte of the extended I/O area (O area)
	OW	0 to 254	a word of the extended I/O area (O area)

Table 3-5 Load and transfer operations/part 2

Operation	Operand		Function
L			Load
	KB	0 to 255	a constant, 1 byte
	KS	2 ASCII characters	a constant, 2 ASCII characters
	KF	-32768 to +32767	a constant as fixed point number
	KG	[1]	a constant as floating point number
	KH	0 to FFFF	a constant as hexadecimal number
	DH	0 to FFFF FFFF	a double word constant as a hexadecimal number
	KM	16-bit pattern	a constant as bit pattern
	KY	0 to 255 for each byte	a constant, 2 bytes
	KT	0.0 to 999.3	a constant timer value (in BCD)
	KC	0 to 999	a constant counter value
	T	0 to 255	a timer, binary coded
	C	0 to 255	a counter, binary coded
LC			Load
	T	0 to 255	a timer
	C	0 to 255	a counter
			in BCD

[1] $\pm 0{,}1469368 \times 10^{-38}$ to $\pm 0{,}1701412 \times 10^{39}$

Load operations

Load operations write the addressed value into ACCU 1. The former contents of ACCU 1 are saved in ACCU 2 (stack lift).

Transfer operations

Transfer operations write the contents of ACCU 1 to the addressed memory location.

Examples of load and transfer operations

Example 1:

Fig. 3-6 illustrates loading/transferring a byte, word or double word from/to a memory area organized in **bytes** (PII, PIQ, flags, I/O).

```
:L IB i    load byte i of the PII into ACCU-1-LL
:L IW j    load bytes j and j+1 of the PII into ACCU-1-L
:L FD k    load flag bytes k to k+3 in ACCU 1
```

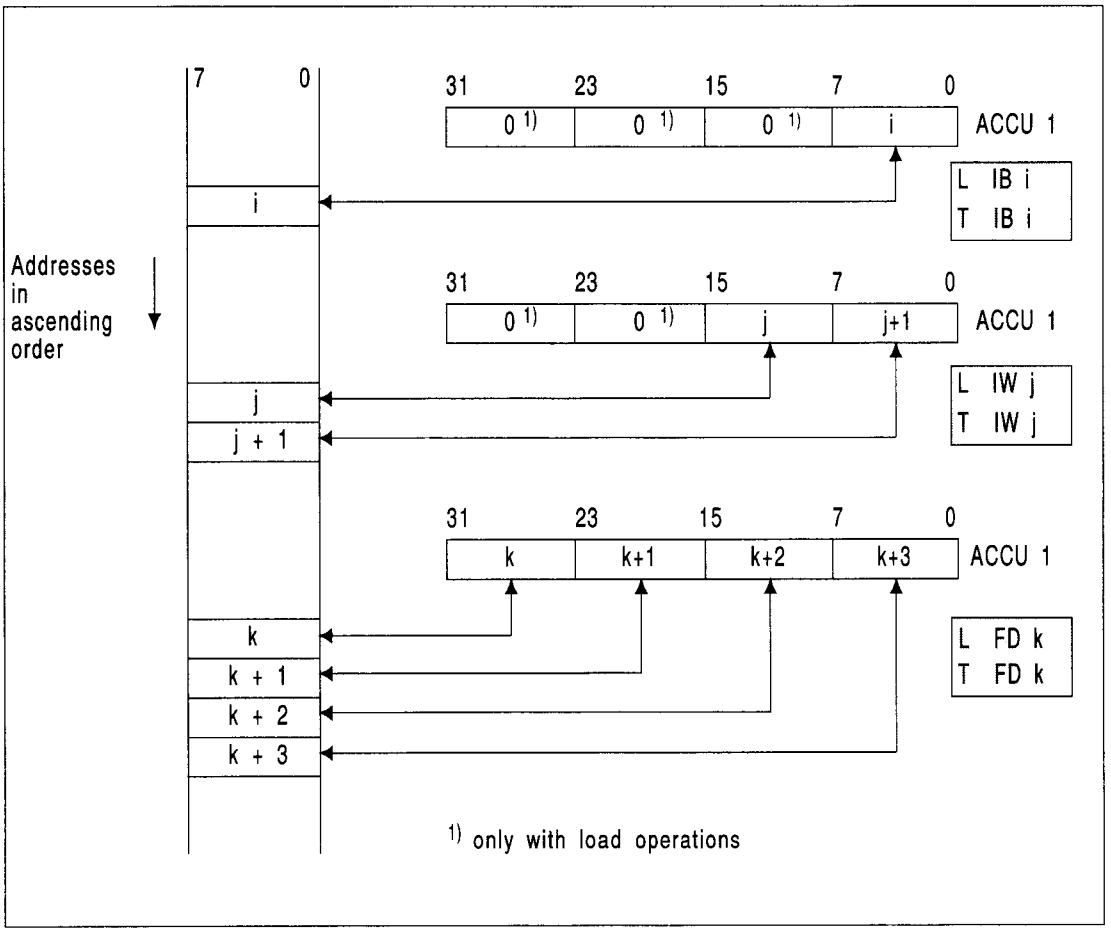


Fig. 3-6 Load and transfer operations in a byte-oriented memory area

Example 2:

Fig. 3-7 illustrates the loading/transfer of a byte, word or double word from/into a memory area organized in **words.**

```
:L DR i load the right byte of data word i into ACCU-1-LL
:L DL j load the left byte of data word j into ACCU-1-LL
:L DW k load data word k into ACCU-1-L
:L DD l load data words l and l+1 into ACCU 1
```

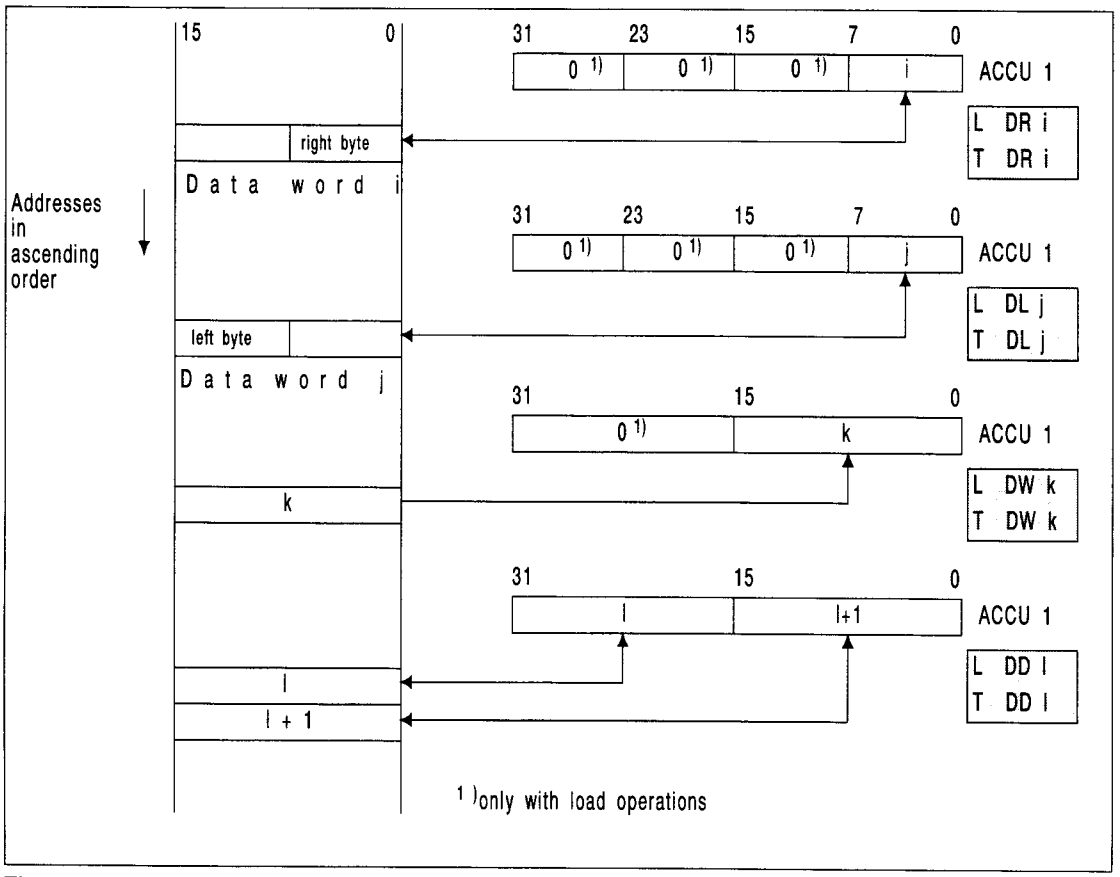


Fig. 3-7 Load and transfer operations in a word-oriented memory area

Note
Load operations do not affect the **condition codes.**
Transfer operations clear the **OS bit.**

When a **byte** or **word** is **loaded** the **extra bits** are **cleared** in ACCU 1.

Addressing I/Os

You can use load and transfer operations to address the I/O peripherals as follows:

- **directly using the following operations:**

  L../T.. ..PY, ..PW, ..OY, ..OW

or

- **using the process image with the following operations:**

  L../T.. ..IB, ..IW, ..ID, ..QB, ..QW, ..QD

  and with logic and set/reset operations

Note
If you use the transfer operations T PY 0 to 127 and T PW 0 to 126, the process output image is updated at the same time. Exception: command output is disabled by the STEP 5 operation BAS (refer to Section 3.5.4).

Note the following points about I/Os:

- A process input/output image exists for 128 input and 128 output bytes of the P peripherals with byte addresses from 0 to 127.
- No process image exists for the entire area of the O peripherals and the P peripherals with relative byte addresses from 128 to 256. (For more information on address space allocation see Section 8.2.2).
- I/O modules with addresses of the O peripherals can only be plugged into expansion units (not in the central controller).
- In **one** expansion unit, you can use either only P peripherals or only O peripherals.



Caution
If you use relative addresses of the O peripherals in an expansion unit, you can no longer use these addresses for I/O modules in the central controller (this would result in double addressing).

Timer and counter operations

To load a timer using a start operation or a counter using a set operation, you must first load the value in ACCU 1.

The following load operations are preferable:

For timers: L KT, L IW, L QW, L FW, L DW, L SW.
For counters: L KC, L IW, L QW, L FW, L DW, L SW.

Starting a **timer** with the selected timer value requires an RLO signal change.

A **counter** is set or started with the selected counter value when a positive-going RLO signal edge is detected.

The following table indicates the signal edge change with corresponding arrows.

Table 3-6 Timer and counter operations

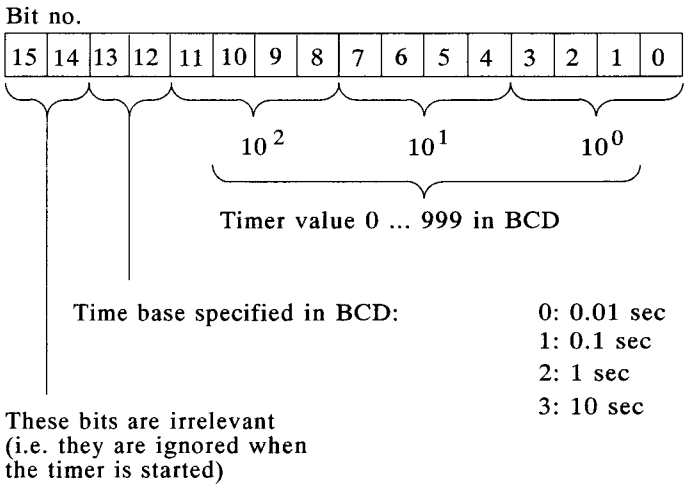
Operation	Operand	RLO 1)	Function
SP	T 0 to 255	↑	Start a timer as a pulse
SE	T 0 to 255	↑	Start a timer as extended pulse
SD	T 0 to 255	↑	Start a timer as ON delay
SS	T 0 to 255	↑	Start a timer as stored ON delay
SF	T 0 to 255	↓	Start a timer as OFF delay
R	T 0 to 255	1	Reset a timer
S	C 0 to 255	↑	Set a counter (BCD number from 0 to 999)
R	C 0 to 255	1	Reset a counter
CU	C 0 to 255	↑	Count up
CD	C 0 to 255	↑	Count down

1) positive-going edge (↑): signal change from '0' to '1'
negative-going edge (↓): signal change from '1' to '0'

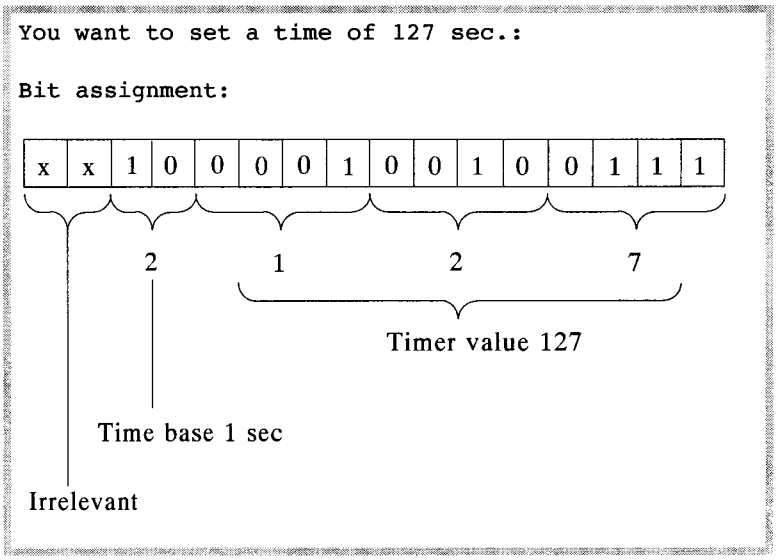
When executing the timer or counter operations SP T, SE T, SD T, SS T, SF T and S C the value in ACCU 1 is transferred to the timer or counter (as with the transfer operation) and the appropriate operation is started.

Timer value

With the operation L KT, you can load a **timer value** directly into ACCU 1 or indirectly from a flag or data word. The value must have the following structure (with L KT, you specify the time base after the period in the operand as shown below):



Example

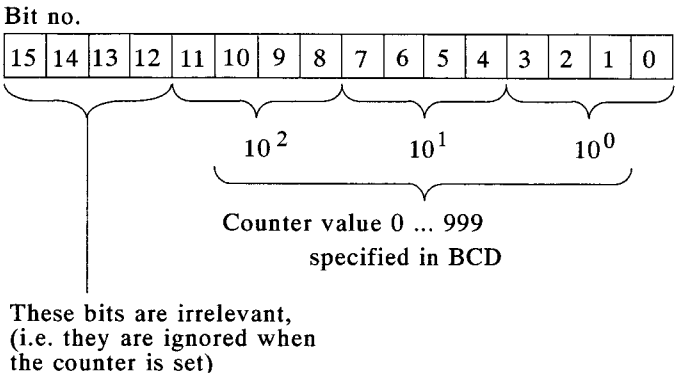


Note
The start of each timer is liable to an inaccuracy of 1 time base! When using timers, you should therefore select the smallest possible time base (time base < timer value):

Example:
time value 4s not: 1 s x 4 inaccuracy: 1 s
but: 0.01 s x 400 inaccuracy: 0.01 s

Counter value

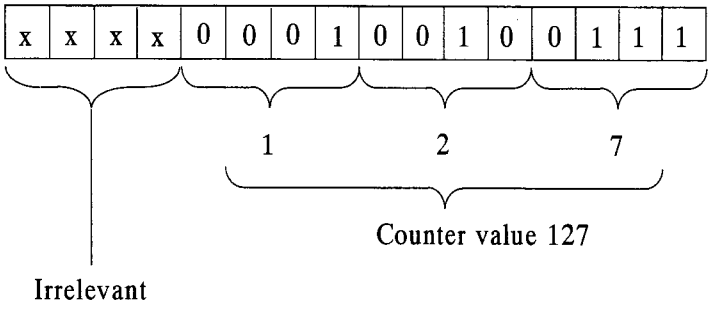
With the operation L KC, you can load a **counter value** directly in ACCU 1 or indirectly from a flag or a data word. The value must have the following structure:



Example

You want to specify a counter value of 127:

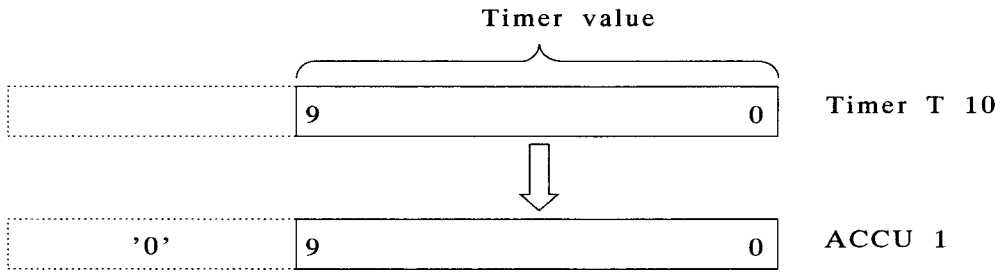
Bit assignment:



In the timer or counter itself, the value is in binary code. If you want to scan the timer or counter, you can load the actual timer or counter value into ACCU 1 **directly** or **in BCD code.**

Further examples of timer and counter values

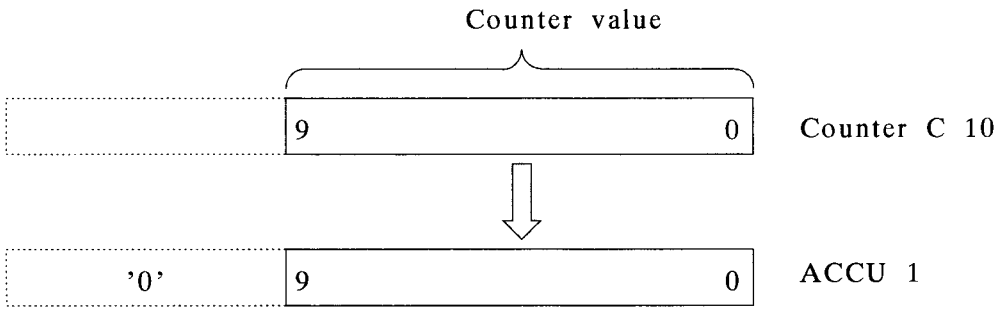
Loading **timer values** directly:



"L T 10": Loads the binary timer value of timer T 10 directly into ACCU 1

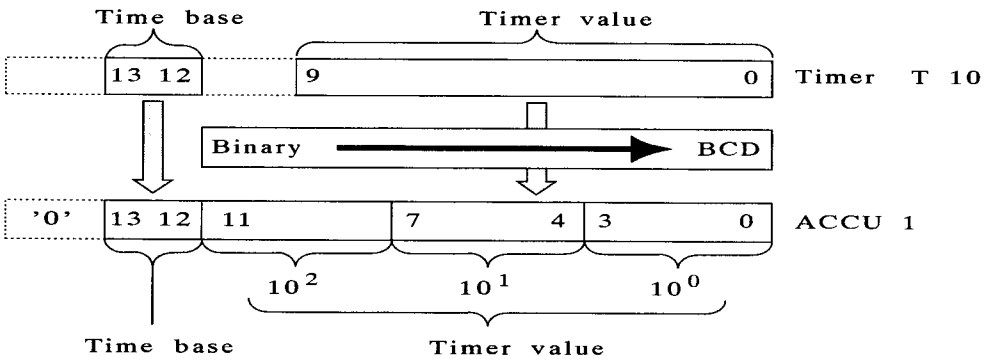
The time base is not loaded.

Loading **counter values** directly:



"L C 10": Loads the binary counter value of counter C 10 directly into ACCU 1

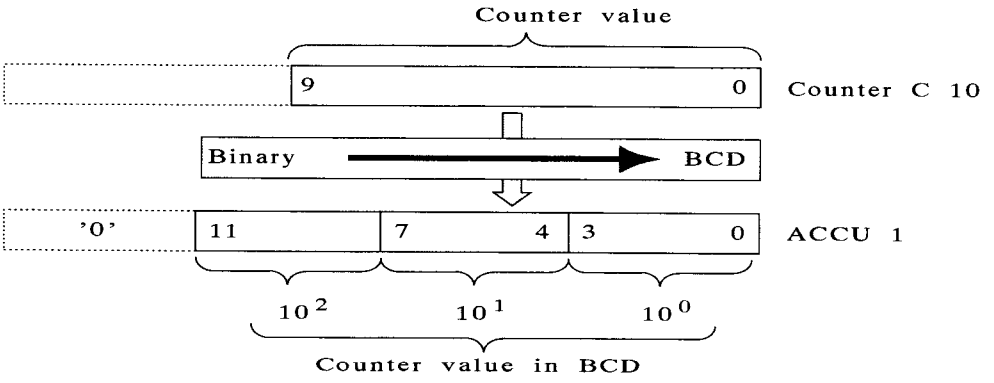
Loading **timer values** in BCD code:



"LC T 10": Loads the timer value and time base of timer T 10 into ACCU 1 in BCD

The time base is also loaded.

Loading **counter values** in BCD code:



"LC C 10": Loads the counter value of counter C 10 into ACCU 1 in BCD

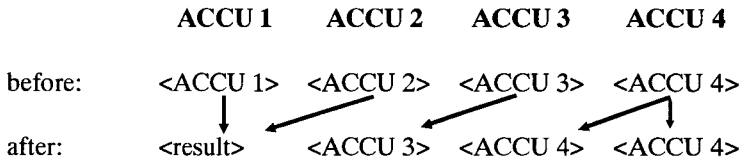
If you load values in BCD, status bits 14 and 15 of the timer or 12 to 15 of the counter are not loaded. They have the value 0 in ACCU 1. The value in the ACCU can now be processed further.

Arithmetic operations

Table 3-7 Arithmetic operations

Operation	Operand	Function
+ F	–	Add two fixed point numbers (16 bits)
- F		Subtract one fixed point number from another (16 bits)
x F		Multiply two fixed point numbers (16 bits)
: F		Divide one fixed point number by another (16 bits): quotient in ACCU-1-L, remainder in ACCU-1-H
+ G		Add two floating point numbers (32 bits)
- G		Subtract one floating point number from another (32 bits)
x G		Multiply two floating point numbers (32 bits)
: G		Divide one floating point number by another (32 bits)

Arithmetic operations logically combine the contents of ACCU 1 and ACCU 2 (e.g. ACCU 2 - ACCU 1). The result is then contained in ACCU 1. An arithmetic operation changes the arithmetic registers as follows (in fixed point operations only the low word):



Note
Within the **supplementary operations**, there are operations for **subtraction** and **addition** of **double word fixed point numbers**.

Comparison operations

Table 3-8 Comparison operations

Operation	Operand	Function
! = >< F > D > = G < <=	–	Compare for equal to Compare for not equal to Compare for greater than Compare for greater than or equal to Compare for less than Compare for less than or equal to ...F: compare two fixed point numbers (16 bits) ...D: compare two fixed point numbers (32 bits) ...G: compare two floating point numbers (32 bits)

Block operations

Table 3-9 Block operations

Operation	Operand	Function
JU JC	 OB 1 to 39 [1)] OB 121 to 255 PB 0 to 255 FB 0 to 255 SB 0 to 255	Jump unconditionally Jump conditionally (only when RLO = 1) to an organization block to a system program special function to a program block to an FB function block to a sequence block
DOU DOC	 FX 0 to 255	Jump unconditionally Jump conditionally (only when RLO = 1) to an FX function block
BE BEC BEU	–	Block end Block end, conditional (only when RLO = 1) Block end, unconditional
C CX	DB 2 to 255 DX 3 to 255	Call a DB data block Call a DX data block
G GX	DB 2 to 255 DX 3 to 255	Generate data block DB Generate data block DX (ACCU 1 must contain the number of data words – maximum 4091 – that the new block is to have)

[1)] only for test purposes!

G DB/GX DX

Generating a data block

The operation G DBx generates a DB data block with the number x ($2 \leq x \leq 255$) in the user memory of the CPU. The content of the data block is **not** assigned the value 0, i.e. the data words can have any contents.
Before programming this statement, you must store the number of data words that the new DB is to have in ACCU-1-L. The operation "G DB" or "GX DX" creates the block header. A data block generated in this way (**without** block header) can occupy a maximum of 4091 words. You can generate longer data blocks using OB 125 (refer to Section 6.5).

If the data block already exists, the length of the DB is not permitted or there is not enough space in the DB-RAM, the system program calls **OB 34**. If this is not loaded, the CPU goes to the stop mode.

The GX DXx operation generates a DX data block in the user memory and is otherwise the same as G DBx.

Note
If the operations G DB/GX DX are used in the START-UP, they must only be executed by the **master CPU**!
If you use these operations, note the information about the special functions OB 124, OB 125 and OB 254/255 at the start of Chapter 6

NOP/display/stop operations

Table 3-10 NOP/display/stop operations

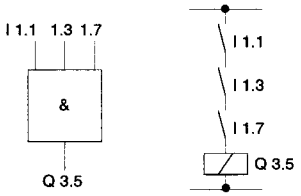
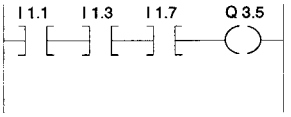
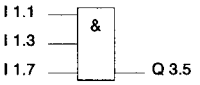
Operation	Operand	Function
NOP 0 NOP 1	–	No operation No operation
BLD	0 to 255 130 131/131/133 255	Display generation operation for the PG: the CPU handles the operation like a no operation Create blank line with carriage return Switch over between STL, CSF, LAD
STP	–	At end of cycle or at end of OB 1, CPU changes to soft STOP.

Note
Since the operation STP is only effective at the end of the cycle, there is no ISTACK entry. The cause of the stoppage is then difficult to find afterwards.
To make diagnosis easier, you should set an identifier before calling the STP operation, e.g. a special bit pattern in a diagnostic DB or use the STEP 5 operation STS (refer to Section 3.5.4).

3.5.2 Programming Examples in the STL, LAD and CSF Methods of Representation

Logic operations

AND operation

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 1.1 A I 1.3 A I 1.7 = Q3.5		

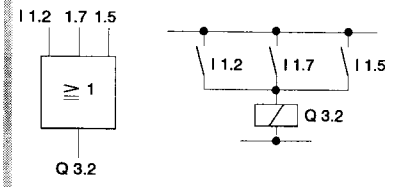
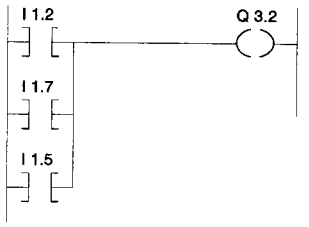
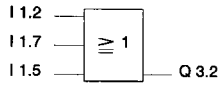
Output Q 3.5 is "1" when all inputs are "1" simultaneously

Output Q 3.5 is "0" if any of the inputs has signal state "0"

The number of scans and the sequence of the logic statements are optional

Logic operations
(continued)

OR operation

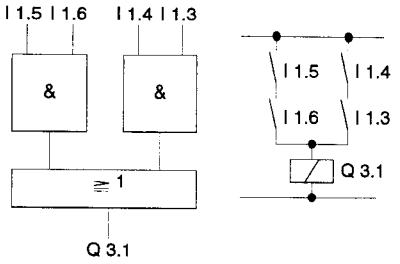
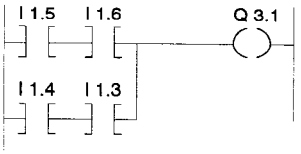
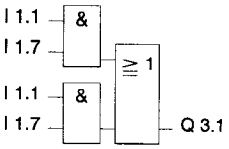
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	O I 1.2 O I 1.7 O I 1.5 = Q3.2		

Output Q 3.2 is "1" when at least one of the inputs is "1"

Output Q 3.2 is "0" when all inputs have the signal state state "0" simultaneously

The number of scans and sequence of programming is optional

AND-before-OR operation

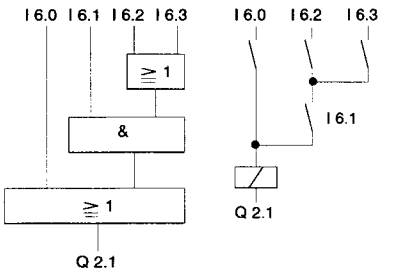
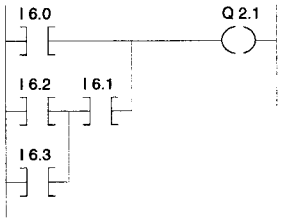
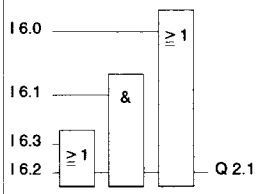
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 1.5 A I 1.6 O A I 1.4 A I 1.3 = Q3.1		

Q 3.1 is "1" when at least one AND condition is satisfied

Q 3.1 is "0" when no AND condition is satisfied

Logic operations
(continued)

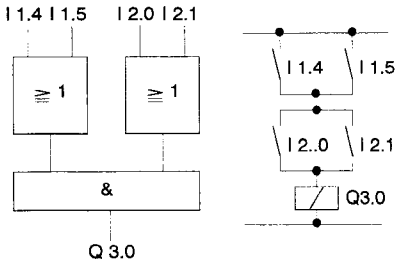
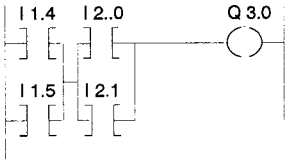
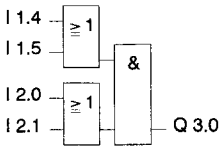
OR-before-AND operation /1st example

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 6.0 O A I 6.1 A (O I 6.2 O I 6.3) = Q2.1		

Output Q 2.1 is "1" when input I 6.0 or input I 6.1 and one of the inputs I 6.2 or I 6.3 has signal state "1"

Output Q 2.1 is "0" when input I 6.0 has signal state "0" and the AND condition is not satisfied

OR-before-AND operation /2nd example

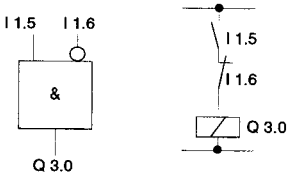
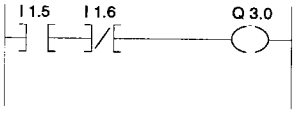
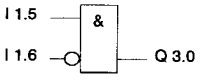
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A (O I 1.4 O I 1.5) A (O I 2.0 O I 2.1) = Q3.0		

Output Q 3.0 is "1" when both OR conditions are satisifed

Output Q 3.0 is "0" when at least one OR condition is not satisfied

Logic operations
(continued)

Scan for signal state "0"

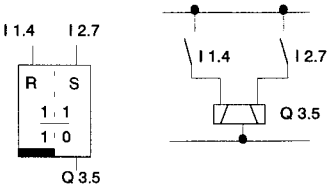
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 1.5 AN I 1.6 = Q3.0		

Output Q 3.0 is "1" only when input I 1.5 has signal state "1" (normally open contact activated) and input I 1.6 has signal state "0" (normally closed contact activated)

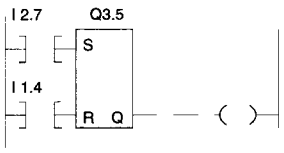
Set/reset operations

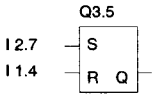
RS flip-flop for a latching signal output

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A   I 2.7
S   Q 3.5
A   I 1.4
R   Q 3.5
```





Signal state "1" at input I 2.7 sets the flip-flop (signal state "1" at output Q 3.5).
If the signal state at input I 2.7 changes to "0", the state of output Q 3.5 is retained (i.e. the signal is latched).

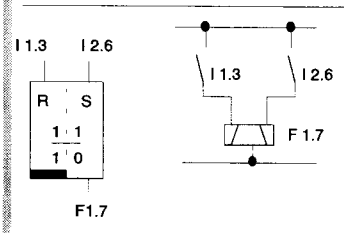
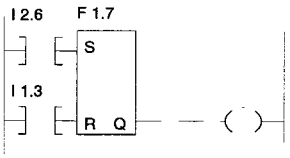
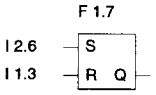
Signal state "1" at input I 1.4 resets the flip-flop (signal state "0" at output Q 3.5).

If the signal state at input I 1.4 changes to "0", the state of Q 3.5 is retained.

When the set signal (input I 2.7) and the reset signal (input I 1.4) are applied at the same time, the scan operation programmed last (in this case A I 1.4) remains in effect for the rest of the program (reset priority).

Set/reset operations
(continued)

RS flip-flop with flags

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 2.6 S F 1.7 A I 1.3 R F 1.7		

Signal state "1" at input I 2.6 sets the flip-flop.

If the signal state at input I 2.6 changes to "0", the signal state of the flag is retained, i.e. the signal is latched.

Signal state "1" at input I 1.3 resets the flip-flop.

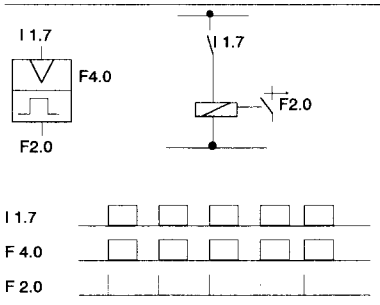
If the signal state at input I 1.3 changes to "0", the signal state of the flag is retained.

When the set signal (input I 2.6) and the reset signal (input I 1.3) are applied at the same time, the scan operation last programmed (in this case A I 1.3) remains in effect for the rest of the program (reset priority).

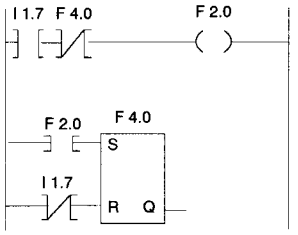
Set/reset operations
(continued)

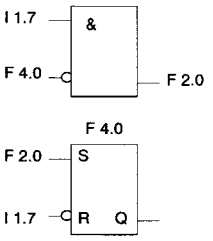
Simulation of a momentary contact relay (one shot)

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A   I 1.7
AN  F 4.0
=   F 2.0
A   F 2.0
S   F 4.0
AN  I 1.7
R   F 4.0
```





On each leading edge of the signal at input I 1.7, the AND condition (A I 1.7 and AN F 4.0) is satisfied; the RLO is "1". This sets flags F 4.0 (edge flag) and F 2.0 (pulse flag).

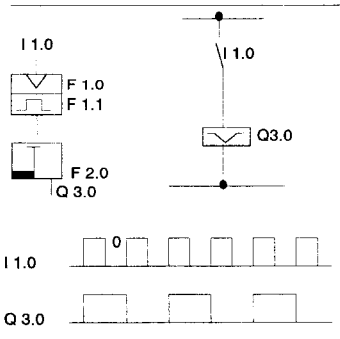
In the next processing cycle, the AND condition A I 1.7 and AN F 4.0 is not satisfied, since flag F 4.0 has already been set.

Flag F 2.0 is reset.

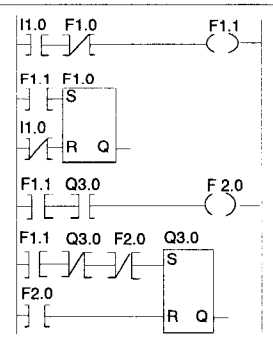
Flag F 2.0 therefore only remains "1" for one program run.

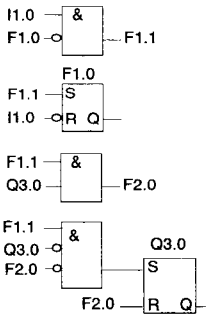
Binary scaler (binary divider)

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A   I 1.0
AN  F 1.0
=   F 1.1
A   F 1.1
S   F 1.0
AN  I 1.0
R   F 1.0
A   F 1.1
A   Q3.0
=   F 2.0
A   F 1.1
AN  Q3.0
AN  F 2.0
S   Q 3.0
A   F 2.0
R   Q 3.0
```



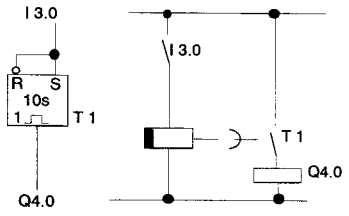


The binary scaler (output Q 3.2) changes its state each time input I 1.0 changes its signal state from 0 to 1 (leading edge). Therefore, only half the input frequency appears at the output of the memory cell.

Timer operations

Pulse timer

Logical/circuit diagram

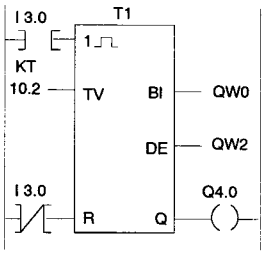


STEP 5 representation

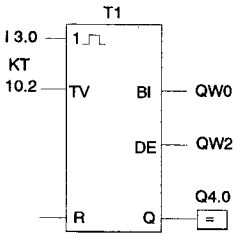
Statement list

```
A    I   3.0
L    KT 10.2
SP   T   1
AN   I   3.0
R    T   1
L    T   1
T    QW  0
LC   T   1
T    QW  2
A    T   1
=    Q   4.0
```

Ladder diagram



Control system flowchart



The timer is started during the first scan if the RLO is "1". Subsequent scans with an RLO of "1" do not affect the timer.

If the RLO is "0", the timer is reset (cleared).

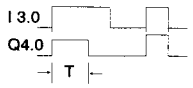
The scan AT or OT produces the signal "1" as long as the timer is running.

KT 10.2:

The timer is loaded with the specified value (10). The number to the right of the decimal point indicates the time base:

0 = 0.1sec 2 = 1sec
1 = 0.1 sec 3 = 10 sec

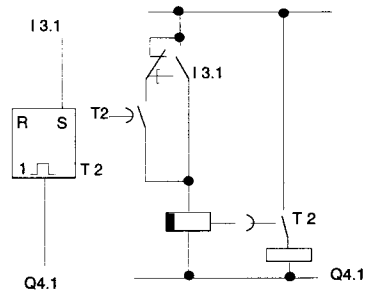
BI and DE are digital outputs of the timer. The time at output BI is in binary code. The time at DE is in BCD code with time base.



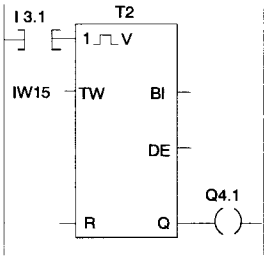
Timer operations (continued)

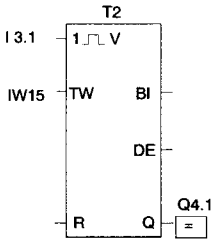
Extended pulse timer

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart



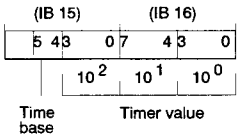

```
A     I   3.1
L     IW 15
SE    T   2
A     T   2
=     Q   4.1
```





The timer is started during the first scan if the RLO is "1".

An RLO of "0" does not affect the timer.



The scan AT or OT produces a signal "1" as long as the timer is running.

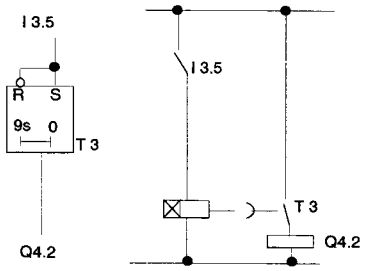
IW 15:
Set the timer with the value of the operand I, Q, F or D in BCD code (in this example, input word 15).



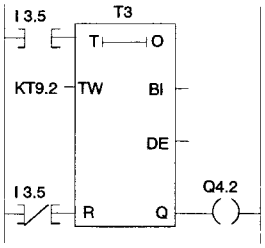
Timer operations (continued)

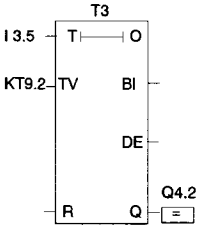
ON-delay timer

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A    I  3.5
L    KT 9.2
SD   T  3
AN   I  3.5
R    T  3
A    T  3
=    Q  4.2
```





The timer is started during the first scan if the RLO is "1". An RLO of "1" during subsequent scans does not affect the timer.

When the RLO is "0", the timer is reset (cleared).

The scan AT or OT produces the signal "1" when the timer has elapsed and the RLO is still applied to the input.

KT 9.2:

The timer is loaded with the specified value (9). The number to the right of the decimal point indicates the time base:

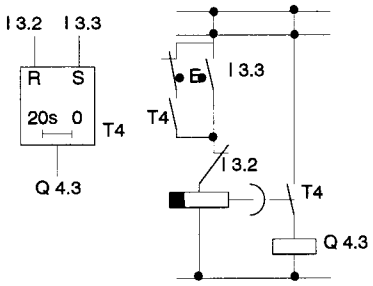
0 = 0.1sec 2 = 10 sec
1 = 0.1 sec 3 = 10 sec



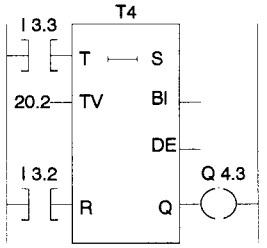
Timer operations (continued)

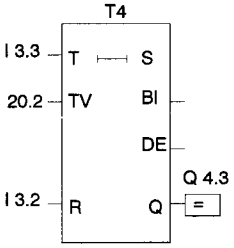
Stored ON-delay timer

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart

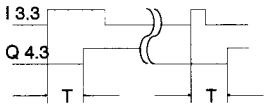



```
A   I   3.3
L   KT  20.2
SS  T   4
A   I   3.2
R   T   4
A   T   4
=   Q   4.3
```



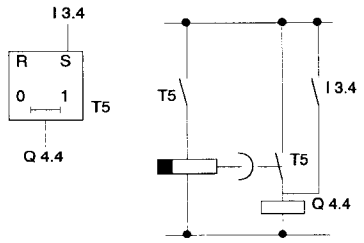


The timer is started during the first scan if the RLO is "1".
An RLO of "0" does not affect the timer.
The scan AT or OT produces the signal "1" when the timer has elapsed. The signal state does not change to "0" until the R T operation resets the timer.

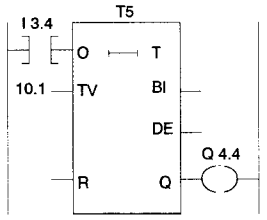


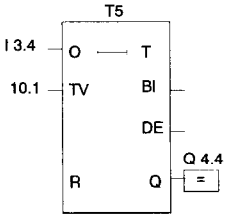
OFF-delay timer

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart

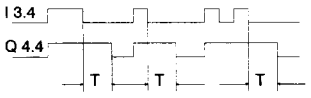



```
A   I   3.4
L   KT  10.1
SF  T   5
A   T   5
=   Q   4.3
```





When the RLO at the start input changes from "1" to "0", the timer is started. It runs for the length of time programmed.
When the RLO is "1", the timer is reset (cleared).

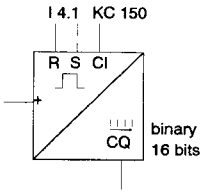


The scan AT or OT produces signal state "1" if the timer is running or the RLO at the input is "1".

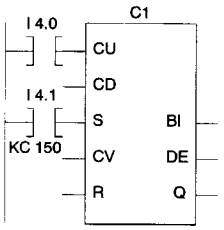
Counter operations

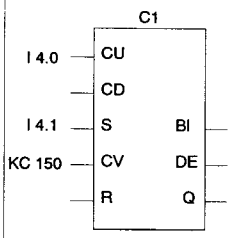
Set counter

Logical/circuit operation	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A   I   4.0
CU  C   1
A   I   4.1
L   KC  150
S   C   1
```



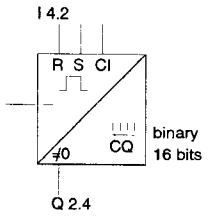


When the result of logic operation changes at the start input (I 4.1) from "0" to "1", the counter is loaded with the specified value (150).

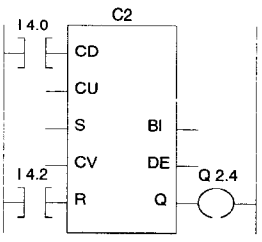
The flag necessary for edge evaluation of the set input is incorporated in the counter word.
BI and DE are digital outputs of the counter cell. The value at BI is in binary code and the value at DE is in BCD.

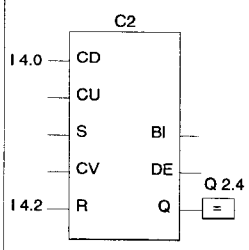
Reset counter

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart




```
A   I   4.0
CD  C   2
A   I   4.2
R   C   2
A   C   2
=   Q   2.4
```



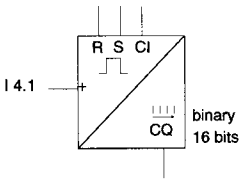
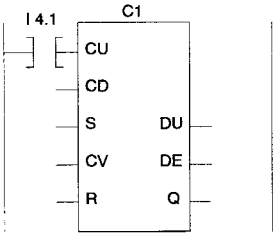
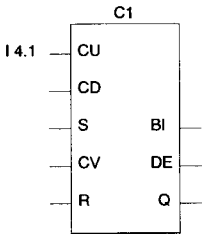


An RLO of "1" (I 4.2) resets the counter to zero.

An RLO of "0" does not affect the counter.

Counter operations
(continued)

Count up

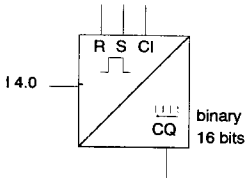
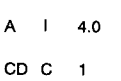
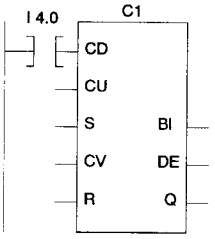
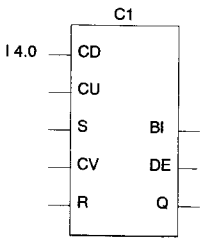
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	A I 4.1 CU C 1		

The value of the addressed counter is incremented by "1" to a maximum value of 999. The function CU is only executed on a positive edge (from "0" to "1") of the logic operation programmed before CU. The flags necessary for edge evaluation of the counter inputs are incorporated in the counter word.

Owing to the two separate edge flags for CU and CD, a counter with two different inputs can be used as an up/down counter.

Counter operations
(continued)

Count down

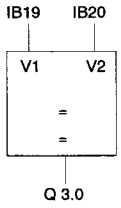
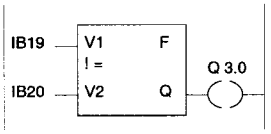
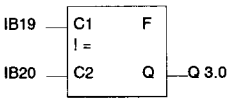
Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
			

The value of the addressed counter is decremented by 1 to a maximum counter value of 0. The function is only executed on a positive edge (from "0" to "1") of the logic operation programmed before the CD. The flags necessary for edge evaluation of the counter inputs are incorporated in the counter word.

Owing to the two separate edge flags for CU and CD, a counter with two different inputs can be used as an up/down counter.

Comparison operations

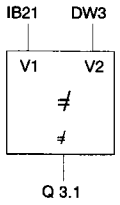
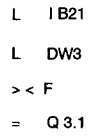
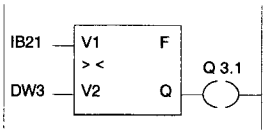
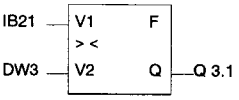
Compare for equal to

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
	L I B19 L IB20 ! = F = Q 3.0		

The first operand is compared with the second operand by the comparison operation. The RLO of the comparison is binary.
RLO = "1": comparison is satisfied if ACCU-1-L = ACCU-2-L
RLO = "0": comparison is not satisfied, when ACCU-1-L is not equal to ACCU-2-L.
The condition codes CC1 and CC0 are set as described in the list of operations.
ACCU-2-H and ACCU-1-H are not involved in the operation for a 16-bit fixed point comparison.
In a 32-bit fixed point comparison (! = D) and floating point comparison (! = G) the entire contents of ACCU 1 and ACCU 2 (32 bits) are compared with each other.
During the comparison, the numerical representation of the operands is taken into account, i.e. the contents of ACCU-1-L and ACCU-2-L are interpreted here as a fixed point number.

Comparison operations
(continued)

Compare for not equal to

Logical/circuit diagram	STEP 5 representation		
	Statement list	Ladder diagram	Control system flowchart
			

The first operand is compared with the second operand by the comparison operation.
The RLO of the comparison is binary.
RLO = "1": comparison is satisfied if ACCU-1-L is not equal to ACCU-2-L.
RLO = "0": comparison is not satisfied if ACCU-1-L equals ACCU-2-L.
The condition codes CC1 and CC0 are set as described at the beginning of Section 3.5.
ACCU-2-H and ACCU-1-H are not involved in the operation for a 16-bit fixed point comparison.
ACCU-2-H and ACCU-1-H are involved in a 32-bit fixed point comparison and floating point comparison.
This information also applies to comparison operations for "greater than", "greater than or equal to", "less than" and "less than or equal to" (see the operations list). During the comparison, the numerical representation of the operands is taken into account, i.e. the contents of ACCU-1-L and ACCU-2-L are interpreted here as a fixed point number.

3.5.3 Supplementary Operations

You can use the supplementary operations set on the programmer only in function blocks (FB and FX). This means that the total operations set for function blocks consists of the basic operations and the supplementary operations.

The system operations also belong to the supplementary functions. You can use the system operations, for example to overwrite the memory at optional locations or to change the contents of the working registers of the CPU. System operations can only be programmed if they have been enabled in the presets menu of the programmer (no longer necessary from S5-DOS Version 2.0 upwards).

If you intend to use system operations, you should be familiar with Chapter 9 "Memory Access Using Absolute Addresses".



Caution
Only experienced system programmers should use the system operations and then only with extreme caution.

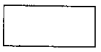
You can only write operations in function blocks in STL. You cannot program function blocks in graphic form (LAD and CSF methods of representation).
This section describes the supplementary operations and covers possible combinations of substitution operations with actual operands.

System operations

System operations are marked in the first column of the tables with S

Binary logic operations

Table 3-11 Binary logic operations with formal operands

Operation	Operand	Function
A =		AND operation, scan a formal operand for signal state '1'
AN =		AND operation, scan a formal operand for signal state '0'
O =		OR operation, scan a formal operand for signal state '1'
ON =		OR operation, scan a formal operand for signal state '0'
		Insert formal operand
		Inputs, outputs, data and flags addressed in binary (parameter types: I, Q; data type BI) and timers and counters (parameter type: T, C) are permitted as actual operands.

Digital logic operations

Table 3-12 Digital logic operations

Operation	Operand	Function
AW		AND operation on the contents of ACCU-1-L and ACCU-2-L
OW		OR operation on the contents of ACCU-1-L and ACCU-2-L
XOW		Exklusive OR operation on the contents of ACCU-1-L and ACCU-2-L

ACCUs 2, 3 and 4 are not affected, however, the condition codes CC 1 and CC 0 are affected (see word condition codes).

Bit test operations

Table 3-13 Bit test operations

Operation	Operand	Function
TB		Scan for signal state "1"
	I 0.0 to 127.7	of an input (PII)
	Q 0.0 to 127.7	of an output (PIQ)
	F 0.0 to 255.7	of a flag
	D 0.0 to 255.15	of a data word bit
	T 0.0 to 255.15	of a timer word bit
	C 0.0 to 255.15	of a counter word bit
	RI 0.0 to 255.15	of a bit in RI area
	RJ 0.0 to 255.15	of a bit in RJ area
	RS 0.0 to 255.15	of a bit in RS area
	RT 0.0 to 255.15	of a bit in RT area
TBN		Scan for signal state "0"
	I 0.0 to 127.7	of an input (PII)
	Q 0.0 to 127.7	of an output (PIQ)
	F 0.0 to 255.7	of a flag
	D 0.0 to 255.15	of a data word bit
	T 0.0 to 255.15	of a timer word bit
	C 0.0 to 255.15	of a counter word bit
	RI 0.0 to 255.15	of a bit in RI area
	RJ 0.0 to 255.15	of a bit in RJ area
	RS 0.0 to 255.15	of a bit in RS area
	RT 0.0 to 255.15	of a bit in RT area

The bit test operations scan the state of the bit and indicate it via the RLO.

Set/reset operations

Table 3-14 Set/reset operations with formal operands

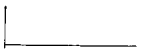
Operation	Operand	Function
S =		Set a formal operand (binary)
RB =		Reset a formal operand (binary)
RD=		Reset a formal operand (digital) for timers and counters
= =		Assign the value of the RLO to a formal operand
		Insert formal operand
		Inputs, outputs and F flags addressed in binary (parameter type: I, Q; data type BI) are permitted as actual operands.

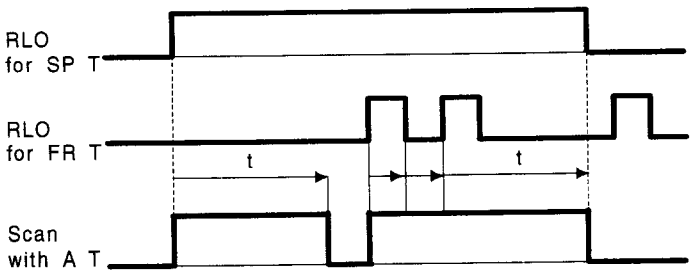
Table 3-15 Set and reset operations

Operation	Operand	Function
SU		Unconditional setting
	I 0.0 to 127.7	of an input (PII)
	Q 0.0 to 127.7	of an output (PIQ)
	F 0.0 to 255.7	of a flag
	D 0.0 to 255.15	of a data word bit
	T 0.0 to 255.15	of a timer word bit
	C 0.0 to 255.15	of a counter word bit
	RI 0.0 to 255.15	of a bit in RI area
	RJ 0.0 to 255.15	of a bit in RJ area
	RS 60.0 to 63.15	of a bit in RS area
	RT 0.0 to 255.15	of a bit in RT area
RU		Unconditional resetting
	I 0.0 to 127.7	of an input (PII)
	Q 0.0 to 127.7	of an output (PIQ)
	F 0.0 to 255.7	of a flag
	D 0.0 to 255.15	of a data word bit
	T 0.0 to 255.15	of a timer word bit
	C 0.0 to 255.15	of a counter word bit
	RI 0.0 to 255.15	of a bit in RI area
	RJ 0.0 to 255.15	of a bit in RJ area
	RS 60.0 to 63.15	of a bit in RS area
	RT 0.0 to 255.15	of a bit in RT area

Timer and counter operations

Table 3-16 Timer and counter operations with formal operands

Operation	Operand	Function
SP =		Start timer specified by the formal operand as a pulse with the value stored in ACCU-1-L (parameter type T).
SD =		Start timer specified by the formal operand as ON delay with the value stored in ACCU-1-L (parameter type T).
SEC =		Start timer specified by the formal operand as extended pulse with the value stored in ACCU-1-L or set counter specified as formal operand with the counter value stored in ACCU-1-L (parameter type: T, C).
SSU =		Start timer specified by the formal operand as stored ON delay with the value stored in ACCU-1-L or increment a counter specified as formal operand (parameter type: T, C).
SFD =		Start timer specified by the formal operand as stored OFF delay with the value stored in ACCU-1-L or decrement a counter specified as formal operand (parameter type: D, C).
FR =		Enable formal operand (timer/counter) for cold restart (see FR T or FR R); (parameter type: T, C).
		Insert formal operand
FR	T 0 to 255	Enable timer for cold restart: The operation is **only** executed on the leading edge of the RLO (change from 0 to 1). The timer is restarted if the RLO is 1 at the time of the start operation. (See timing diagram below the table).
	C 0 to 255	Enable a counter for setting or resetting: The operation is executed **only** on the leading edge of the RLO (change from 0 to 1). The counter is only started if the RLO = 1 at the time of the start operation.



Examples

a)

Function block call	Program in the function block	Program executed
:JU FB 203 NAME :EXAMPLE1 ANNA : I 10.3 BERT : T 17 JOHN : Q 18.4	:A =ANNA :L KT 010.2 :SSU =BERT :A =BERT := =JOHN	:A I 10.3 :L KT 010.2 :SS T 17 :A T 17 := Q 18.4

b)

:JU FB 204 NAME :EXAMPLE2 MAXI : I 10.5 IRMA : I 10.6 EVA : I 10.7 DORA : C 15 EMMA : F 58.3	:A =MAXI :SSU =DORA :A =IRMA :SFD =DORA :A =EVA :L KC 100 :SEC =DORA :AN =DORA := =EMMA	:A I 10.5 :CU C 15 :A I 10.6 :CD C 15 :A I 10.7 :L KC 100 :S C 15 :AN C 15 := F 58.3

c)

:JU FB 205 NAME :EXAMPLE3 BILL : I 10.4 JACK : T 18 EGON : IW 20 YOGI : F 100.7	:A =BILL :L =EGON :SEC =JACK :A =JACK := =YOGI	:A I 10.4 :L IW 20 :SE T 18 :A T 18 := F 100.7

Load and transfer operations

Table 3-17 Load and transfer operations with formal operands

Operation	Operand	Function
L =		Load a formal operand: The value of the operand specified as a formal operand is loaded into the ACCU (parameter type: I, T, C, Q; data type: BY, W, D).
LCD =		Load a formal operand in BCD code: The value of the timer or counter specified as a formal operand is loaded into the ACCU in BCD code (parameter type: T, C).
LW =		Load the bit pattern of a formal operand: The bit pattern of a formal operand is loaded into the ACCU (parameter type: D; data type: KF, KH, KM, KY, KS, KT, KC).
LWD =		Load the bit pattern of a formal operand: The bit pattern of a formal operand is loaded into the ACCU (parameter type: D; data type: KG).
T =		Transfer to a formal operand: The contents of the accumulator are transferred to the operand specified as a formal operand (parameter type: I, Q; data type: BY, W, D).
		Insert formal operand

Actual operands permitted include those of the corresponding basic operations **except for S flags**. For the "LW=" operation, permissible data types include a binary pattern (KM) or a hexadecimal pattern (KH), two absolute numbers of 1 byte each (KY), a character (KS), a fixed point number (KF), a timer value (KT) and a counter value (KC). For "LWD=" permissible data is a floating point number.

Table 3-18 Load and transfer operations with special operands

Operation	Operand	Function
L	RI 0 to 255	Load a word from the interface data area into ACCU 1 (RI area)
	RJ 0 to 255	Load a word from the extended interface area into ACCU 1 (RJ area)
L	RS 0 to 255	Load a word from the system data area into ACCU 1 (RS area)
	RT 0 to 255	Load a word from the extended system data area into ACCU 1 (RT area)
T	RI 0 to 255	Transfer the contents of ACCU 1 to a word in the interface data area (RI area)
	RJ 0 to 255	Transfer the contents of ACCU 1 to a word in the extended interface data area (RJ area)
T	RS 60 to 63	Transfer the contents of ACCU 1 to a word in the system data area (RS area)
	RT 0 to 255	Transfer the contents of ACCU 1 to a word in the extended system data area (RT area)

In contrast to the RI, RJ and RT areas, you can only use words RS 60 to RS 63 of the RS area. Refer to Section 8.3.4 "RS/RT Area".

You can use the RT area in its complete length (RT 0 to RT 255) providing you do not use any standard function blocks.

Arithmetic operations

Table 3-19 Arithmetic operation ENT

Operation	Operand	Function
ENT	–	This causes a stack lift into ACCUs 3 and 4: <ACCU 4> := <ACCU 3> <ACCU 3> := <ACCU 2> <ACCU 2> := <ACCU 2> <ACCU 1> := <ACCU 1> ACCUs 1 and 2 are not changed. The old contents of ACCU 4 are lost.

Example

```
The following fraction must be calculated: (30 + 3 * 4) / 6 = 7
```

	ACCU 1	ACCU 2	ACCU 3	ACCU 4
Contents of the ACCUs before the sequence of arithmetic operations	a	b	c	d
L KF +30	30	a	c	d
L KF +3	3	30	c	d
ENT	3	30	30	c
L KF +4	4	3	30	c
x F	12	30	c	c
+ F	42	c	c	c
L KF +6	6	42	c	c
: F	7	c	c	c

Table 3-20 Supplementary arithmetic operations

Operation		Operand		Function
S	ADD	BN	-128 to +127	Add a byte constant (fixed point) to ACCU-1-L (includes sign change)/the condition code in CC 0, CC 1, OV and OS are not affected! – ACCU-1-H and ACCUs 2 to 4 remain unchanged.
S	ADD	KF	-32 768 to +32 767	Add a fixed point constant (word) to ACCU-1-L/ the condition codes in CC 0, CC 1, OV and OS are not affected! – ACCU-1-H and ACCUs 2 to 4 remain unchanged.
S	ADD [1]	DH	0000 0000 to FFFF FFFF	Add a double word fixed point constant to ACCU 1/the condition codes in CC 0, CC 1, OV and OS are not affected! – ACCUs 2 to 4 remain unchanged.
S	+D [1]			Add two double word fixed point constants (ACCU 2 + ACCU 1)/the result can be evaluated in CC 0/CC 1. [2]
S	-D [1]			Subtract two double word fixed point constants (ACCU 2 - ACCU 1)/the result can be evaluated in CC 0/CC 1. [2]
S	TAK			Swap the contents of ACCU 1 and ACCU 2

[1] Programming is dependent on the PG type and the release of the PG system software.

[2] For changes in ACCU 2 and ACCU 3: see Section 3.5.1 "Basic Operations/Arithmetic Operations".

3.5.4 Executive Operations

The executive operations also include system operations.



Caution
System operations should only be used with great care and then only by experienced programmers familiar with the system.

System operations are indicated in the table by **S**

Jump operations

When you use the supplementary jump operations, you indicate the jump destination for unconditional jumps symbolically. The symbolic parameter of the jump operation is identical to the symbolic address of the destination statement. When programming, remember that the absolute jump distance should not exceed ± 127 words and a STEP 5 statement can consist of more than one word. You can only execute these jumps within a block; jumps over segment boundaries are not permitted ("segment" = structural element in PBs, SBs, FBs, FXs and OBs; see PG description).

Note
The jump statement and jump destination (symbolic address) must be in the same segment. A symbolic address can only be used **once** per segment.
Exception: this does not apply to the JUR jump for which you specify an absolute jump distance as the parameter.

Table 3-21 Jump operations

Operation		Operand	Function
JU	=	addr (addr =symbolic address with maximum 4 characters)	Jump unconditionally: The jump is executed regardless of conditions
JC	=		Jump conditionally: the conditional jump is executed only if the RLO is 1. If the RLO is 0, the statement is not executed and the RLO is set to 1.
JZ	=		Jump if result is '0' : the jump is executed only if CC 1 is 0 and CC 0 is 0. The RLO is not changed.

Table 3-21 continued:

Operation		Operand	Function
	JN =	addr (addr = symbolic address with maximum 4 characters)	Jump if result is not 0 : the jump is executed only if CC1 is not equal to CC0. The RLO is not changed.
	JP =		Jump if result > '0' : the jump is only executed if CC 1 = 1 and CC 0 = O. The RLO is not changed.
	JM =		Jump if result < '0': the jump is only executed if CC 1 = 0 and CC 0 = 1. The RLO is not changed.
	JO =		Jump on overflow: the jump is executed when the OV condition code is 1. If there is no overflow (OV is 0), the jump is not executed. The RLO is not changed. An overflow occurs when an arithmetic operation exceeds the permissible range for a given numerical representation.
	JOS =		Jump when the OS (stored overflow) condition code is set: the jump is executed when the condition code OS is 1. If there is no overflow (OS is 0), the jump is not executed. The RLO is not changed. An overflow occurs when an arithmetic operation exceeds the permissible range for a given numerical representation.
S	JUR	-32 768 to +32 767	Relative jump within the user memory or within a function block (e.g. to arrive in a different segment). The operation is always executed regardless of conditions. The operand is the number of words difference between the address of the jump destination - the current destination. The jump is executed either to a higher (positive operand) or lower (negative operand) address than the current operation.



Caution

If you use **JUR** incorrectly, undefined statuses can occur in the system. It should only be used by extremely experienced programmers with detailed knowledge of the system.

Shift operations

Table 3-22 Shift operations

Operation	Operand	Function (operation with ACCU 1)
SLW	0 to 15	Shift a word to the left (vacant positions to the right are padded with zeros)
SRW	0 to 15	Shift a word to the right (vacant position to the left are padded with zeros)
SLD	0 to 32	Shift a double word to the left (vacant positions to the right are padded with zeros)
SSW	0 to 15	Shift a word with sign to the right (vacant positions to the left are padded with the sign - bit 15)
SSD	0 to 32	Shift a double word with sign to the right (vacant positions to the left are padded with the sign - bit 31)
RLD	0 to 32	Rotate to the left
RRD	0 to 32	Rotate to the right

Only ACCU 1 is involved in the execution of shift operations. The parameter part of these operations specifies the number of positions by which the accumulator contents should be shifted or rotated. For the SLW, SRW and SSW operations, only the low word of ACCU 1 is involved in the shift operations. For SLD, SSD, RLD and RRD operations, the entire contents of ACCU 1 (32 bits) are involved.

Shift operations are executed regardless of conditions.

You can use jump operations to scan the value of the last bits shifted out using CC 1/CC 0.

Shift: last bit shifted	CC 1	CC 0	Jump operation
0	0	0	JZ=
1	1	0	JN= JP=

Examples

1. You want to shift the contents of data word DW 52 four bits to the left and write them to data word DW 53.

```
STEP 5 program:    Contents of the data words:

:L    DW 52        KH = 14AF
:SLW  4
:T    DW 53        KH = 4AF0
```

2. You want to read the input double word ID 0, and shift the contents of ACCU 1 so that the bit positions of the input double word shown in bold face are retained and the remaining bit positions are set to defined values (0H or 0FH).

```
STEP 5 program:    Contents of ACCU 1 (hexadecimal)

                   ACCU-1-H:          ACCU-1-L:

:L    ID 0         2348               ABCD
:SLW     4         2348               BCD0
:SRW     4         2348               0BCD
:SLD     4         3480               BCD0
:SSW     4         3480               FBCD
:SSD     4         0348               0FBC
:RLD     4         3480               FBC0
:RRD     4         0348               0FBC
```

3. Application: Multiplication by the 3rd power, e.g. new value = old value x 8

```
:L    FW 10
:SLW     3
:T    FW 10        Caution: do not exceed the
                            positive area limit!
```

4. Application: Division by the 2nd power, e.g. new value = old value : 4

```
:C    DB 5
:L    DW 0
:SRW     2
:T    DW 0
```

Conversion operations

Table 3-23 Conversion operations

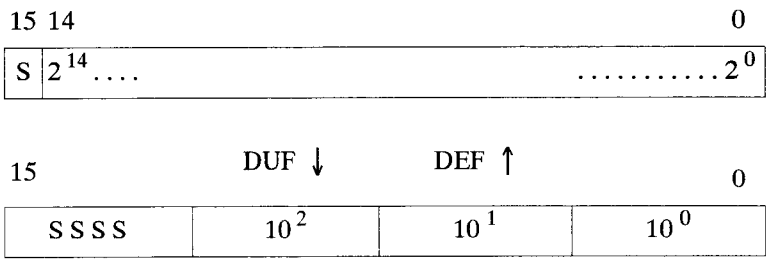
Operation		Function
	CFW	Form the 1's complement of ACCU-1-L (16 bits)
	CSW	Form the 2's complement of ACCU-1-L (16 bits)
	CSD	Form the 2's complement of ACCU 1 (32 bits)
	DEF	Convert a fixed point number (16 bits) from BCD to binary
	DUF	Convert a fixed point number (16 bits) from binary to BCD
	DED	Convert a double word (32 bits) from BCD to binary
	DUD	Convert a double word (32 bits) from binary to BCD
	FDG	Convert a fixed point number (32 bits) to a floating point number (32 bits)
	GFD	Convert a floating point number to a fixed point number (32 bits)

DEF

The value in ACCU-1-L (bits 0 to 15) is interpreted as a BCD number. After the conversion, ACCU-1-L contains a 16-bit fixed point number.

DUF

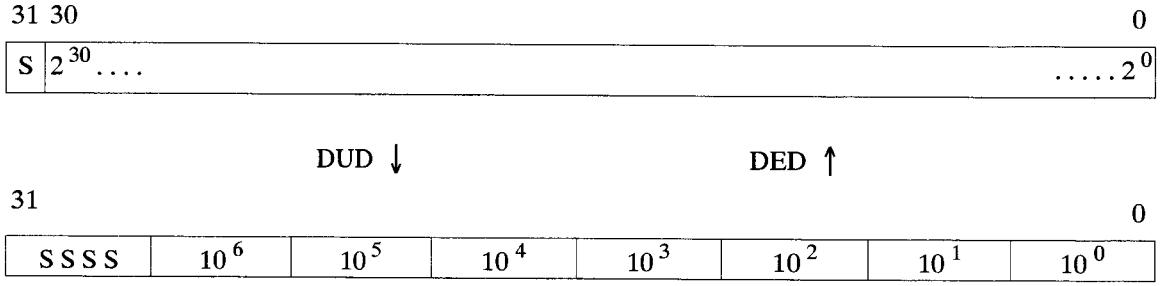
The value in ACCU-1-L (bits 0 to 15) is interpreted as a 16-bit fixed point number. After the conversion, ACCU-1-L contains a BCD number.



S (sign): 0 = positive
1 = negative

DED The value in ACCU 1 (bits 0 to 31) is interpreted as a BCD number. After the conversion, ACCU 1 contains a 32-bit fixed point number.

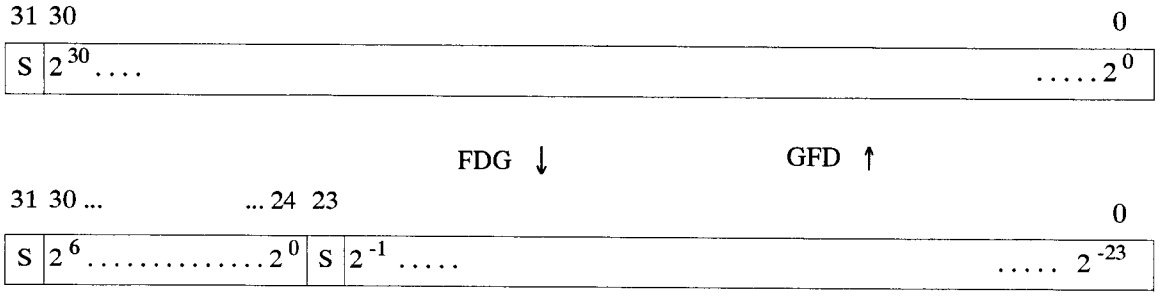
DUD The value in ACCU 1 (bits 0 to 31) is interpreted as a 32-bit fixed point number. After the conversion, ACCU 1 contains a BCD number.



S (sign): 0 = positive
1 = negative

FDG The value in ACCU 1 (bits 0 to 31) is interpreted as a 32-bit fixed point number. After the conversion, ACCU 1 contains a floating point number (exponent and mantissa).

GFD The value in ACCU 1 (bits 0 to 31) is interpreted as a floating point number. After the conversion, ACCU 1 contains a 32-bit fixed point number.



The conversion is made by multiplying the (binary) mantissa by the value of the (binary) exponent by shifting the mantissa value to more significant bits past an imaginary decimal point by the value of the exponent (base 2). After the multiplication, remnants of the original mantissa remain to the right of the imaginary decimal point. These bit places are cut off from the whole result.

This conversion algorithm produces the following result classes:

- Floating point numbers ≥ **0 or ≤ -1** result in the **next lower number**.
- Floating point numbers **< 0 and > -1** result in the **value '0'**.

Conversion examples

Floating point number GFD		32-bit fixed point number
+5,7	→	5
-2,3	→	-3
-0,6	→	**0**
+0,9	→	**0**

Examples of CFW, CSW

1. You want the contents of data word DW 64 inverted bit for bit (reversed) and stored in data word DW 78.

```
STEP 5 program:     Assignment of the data words:

:L DW 64            KM = 0011111001011011
:CFW
:T DW 78            KM = 1100000110100100
```

2. The contents of data word DW 207 are interpreted as a fixed point number and stored in data word 51 with a reversed sign.

```
STEP 5 program:     Assignment of the data words:

:L DW 207           KF = +51
:CSW
:T DW 51            KF = -51
```

Decrement/ increment

Table 3-24 Decrement/increment operation

Operation	Operand	Function
D	1 to 255	Decrement the low byte (bits 0 to 7) of ACCU-1-L by the value of the operand [1]
I	1 to 255	Increment the low byte (bits 0 to 7) of ACCU-1-L by the value of the operand [1]

[1] The contents of the low byte of ACCU-1-L are decremented or incremented by the number specified as the operand without a carry. The operation is executed regardless of conditions.

Example

```
STEP 5 program:     Assignment of the data words:

   :L DW 7          KH = 1010
   :I   16
   :T DW 8          KH = 1020
   :D   33
   :T DW 9          KH = 10FF
```

Processing operations

Table 3-25 Processing operations

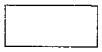
Operation	Operand	Function
DO	DW 0 to 255	Process data word: the following operation is combined with the parameter specified in the address data word and executed.
	FW 0 to 254	Process flag word: the following operation is combined with the parameter specified in the addressed F flag and executed.
DO =		Process formal operand (parameter type B): Only C DB, JU PB, JU OB, JU FB, JU SB can be substituted.
		Insert formal operand

Table 3-25 continued:

Operation		Operand	Function
S	BI	[1]	Indirect processing of a formal operand: execute an operation whose operation code is stored in a formal operand. The number of the formal operand must be stored in ACCU 1.
	B	RS 60 to 63 [1]	Execute an operation whose operation code is stored in the system data area (RS = free system data: RS 60 to 63). In 2-word operations the 2nd word must be loaded in RS n + 1.

[1] The value in the formal operand or system data is interpreted as the operation code of a STEP 5 operation and is then executed.

Note
Only the following operations can be combined with **DO DW**, or **DO FW**, **DI** or **DO RS**:

- A.. , AN.. , O.. , ON.. , S.. , R.. , =..
  with areas I, Q, F, S,
- FR T, R T, SF T, SD T, SP T, SS T, SE T,
- FR C, R C, S C, CD C, CU C,
- L.., T.. with areas P, O, I, Q, F, S, D, RI, RJ, RS, RT,
- L T, L C,
- LC T, LC C,
- JU=, JC=, JZ=, JN=, JP=, JM=, JO=,
- SLW, SRW,
- D, I, SED, SEE,
- C DB, JU.. , JC.., G DB, GX DX, CX DX, DOC FX, DOU FX.

The PG does not check the legality of the combinations!

Examples of DO operations

DO DW/DO FW

Operand substitution

Using the statements "DO DW" and "DO FW" you can access data with a substitution, e.g. in a program loop. The substituted access consists of the statement DO DW/DO FW followed immediately by one of the STEP 5 operations listed above.
"Substituted" means that the operand for the operation is not programmed as a static value but is fixed during the course of the STEP 5 program.

Select the operand **type** from the range permitted for the operation when you write your program, e.g. **PB** for the operation "JU PB nn":

You must first load the operand **value** (**nn** in the example) in a data word or F flag word (parameter word) before the substituted access with DO DW/DO FW.

1. Principle of substitution:

```
          :L    KF +120
          :T    FW 14       load FW 14 with the value "KF  +120"
          :DO   FW 14
          :L    IB 0
                |
                |_________  before the operation "L IB" is executed,
                            the operand value '0' is replaced by value '120';
                            Operation executed: L IB 120
```

2. Data word as index register:

The contents of data words DW 20 to DW 100 are set to signal state '0'. The index register for the parameter of the data words is DW 1.

```
          :L    KF +20      supply the index register
          :T    DW 1
  M001    :L    KF +0       reset
          :DO   DW 1
          :T    DW 0
          :L    DW 1        increment the index register
          :L    KF +1
          :+F
          :T    DW 1
          :L    KF +100
          :<=F
          :JC   =M001       jump if the index is within the range
          ...               remaining STEP 5 program
```

Continued on next page

Examples of operand substitution continued:

3. **Jump distributor for subroutine techniques:**

```
           :DO     FW 5
        ┌──:JU     =M001   Contents of flag word FW 5:
+       │  :JU     =M002
Jump    │  :JU     =M003          jump distance
distance┘  :JU     =M004          (maximum ± 127)
           :JU     =M005
           : .
           : .
  M001     : .
           : .
           :BEU
  M002     : .             Advantage:
           : .             all program sections are
           :BEU            contained in one block.
  M003     : .
           : .
           :BEU
```

4. **Jump distributor for block calls:**

```
           :DO   FW 10                    Contents of flag word FW 10:
           :JU   PB 0 ┬──────PB 0
                      ├──────PB 1          Block no. x
                      ├──────PB 2
                      ├──────PB 3
                      ├──────.
                      ├──────.
                      └──────PB x
```

Operand substitution with binary operations

For operand substitutions with binary operations you can use the following operand types: inputs, outputs, F flags, S flags, timers and counters.
In this substitution, the structure of the F flag word or data word (parameter word) depends on the operation you are using.

Parameter word for inputs and outputs

Bit no.	15 11	10 8	7	6 0
	no significance	Bit address from 0 to 7	0	Byte address from 0 to 127

Parameter word for F flags

Bit no.	15 11	10 8	7 0
	no significance	Bit address from 0 to 7	Byte address from 0 to 255

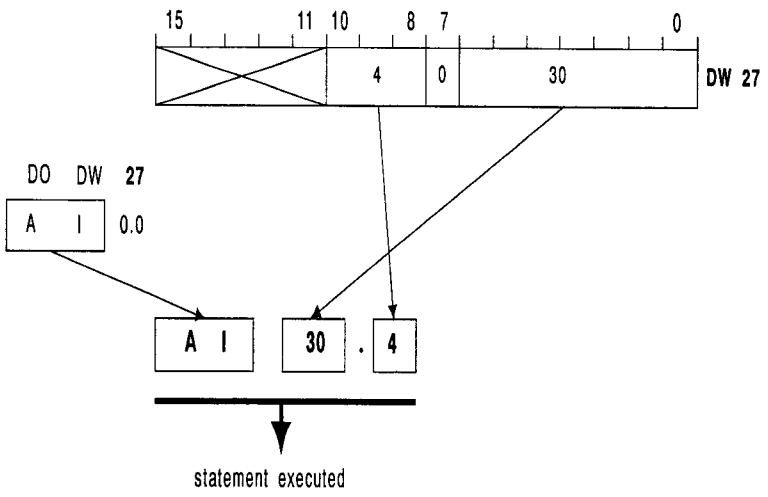
Parameter word for S flags

Bit no.	15	14 12	11 0
	0	Bit address from 0 to 7	Byte address from 0 to 4095

Parameter word for timers and counters

Bit no.	15 8	7 0
	no significance	Number of timer or counter cell from 0 to 255

Principle of the substitution with a binary operation



Example of DI operation

In function block FB 1, STEP 5 operations are executed whose operation codes were transferred by a calling block as formal operands FW 10, FW 12 and FW 14.
Which of the operation codes is executed is written by the calling block as a consecutive number in flag word **FW 16.**
The result of the executed operation is then entered in ACCU 1 and is transferred to flag word FW 18.

```
FB 1

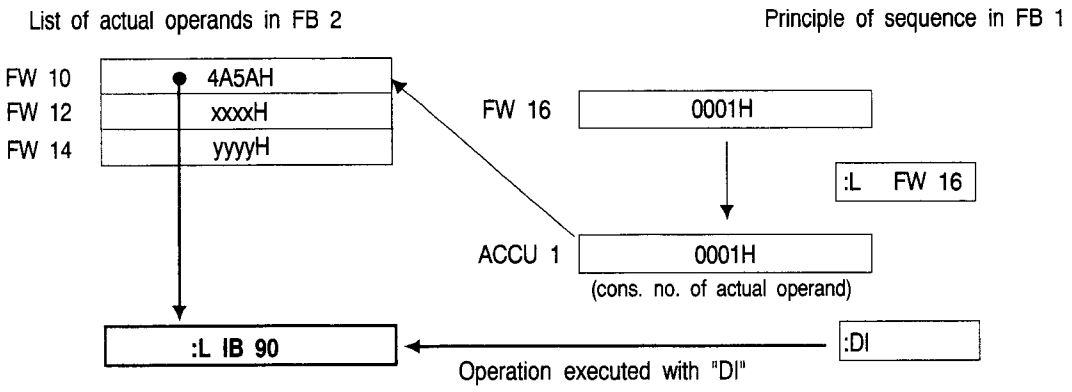
NAME  :TEST

DECL  :FW10      I/Q/D/B/T/C: D  KM/KH/KY/KS/KF/KT/KC/KG:  KH
DECL  :FW12      I/Q/D/B/T/C: D  KM/KH/KY/KS/KF/KT/KC/KG:  KH
DECL  :FW14      I/Q/D/B/T/C: D  KM/KH/KY/KS/KF/KT/KC/KG:  KH

      :L    FW 16          cons. number of formal operand
      :                    with required operation code
      :DI                  transferred operation code is executed
      :T    FW 16          result from ACCU 1
      :BE

FB 2

      :
      :L    KF +1
      :T    FW 16          cons. no. of formal operand with operation code
      :JU   =AUFR
      :
      :
AUFR  :
      :JU   FB 1           call FB TEST
NAME  :TEST
FW10  :     KH 4A5A        op. code "L IB 90",          formal operand 1
FW12  :     KH xxxx        other operation code,        formal operand 2
FW14  :     KH yyyy        other operation code,        formal operand 3
      :T    FW 18          ACCU 1 → FW 18
      :BE
```



I/O operations

Table 3-26 I/O operations

Operation	Operand	Function
IA		Disable process interrupt servicing (via IB 0) (not for system interrupts!)
RA		Enable process interrupt servicing (via IB 0) (not for system interrupts!)
IAE		Disable addressing error
		Enable addressing error
RAE		Disable command output: PIQ is no longer influenced by the operations S Q, R Q, = Q, T PY and T PW.
BAS		Enable command output
BAF		

"Enable/disable process interrupts" can, for example, be used when process interrupt driven processing is to be suppressed during time-controlled processing. In the program section between the IA and RA statements, no process interrupt driven processing is possible. Note the special function OB 122 "disable interrupts", Section 6.3.

Other operations

Table 3-27 Other operations

Operation	Operand	Function
S STS		Stop command leading directly to a soft STOP.
STW		Stop command leading directly to a hard STOP. (state can only be exited with POWER DOWN/UP).
SIM		Set interrupt mask (UAMW) (32 bits): before calling the operation, the bit pattern for the mask must be loaded in ACCU 1 (32 bits)
LIM		Read interrupt mask: bit pattern of the interrupt mask (32 bits) is loaded in ACCU 1

SIM/LIM – set/read interrupt condition code mask (UAMW)

The interrupt mask "masks" interrupts in the interrupt condition code word until the end of the cycle, i.e. all interrupts remain pending, but running of the program is not interrupted by them.

Bit in the interrupt condition code mask = 0: interrupt disabled
Bit in the interrupt condition code mask = 1: interrupt enabled

Meaning of the bits in UAMW-H or ACCU-1-H:

15							8	7							0
INTA	INTE	INTF	INTG	WEFE	WA	PA	BULE	PEU	HALT	ES	AV	INTAS	TAU	DARY	KZU

Meaning of the bits in UAMW-L or ACCU-1-L:

15							8	7							0
–	KB	KDB	STS	TLAF	SUF	STUEB	STUEU	NAU	ZA	QVZ	ADF	PARE	ZYK	STOP	HOLD

Table 3-28 Meaning of the abbreviations in UAMW

Abbrev.	**Meaning**
	High word
INTA	S5 bus/system interrupt A
INTE	S5 bus/system interrupt E
INTF	S5 bus/system interrupt F
INTG	S5 bus/system interrupt G
WEFE	Collision of timed interrupts
WA	Timed interrupt
PA	Process interrupt
BULE	Bus lock error
PEU	I/Os not ready
HALT	Stop instruction from coordinator COR
ES	Single step mode
AV	Address comparison active
INTAS	Interrupt from SPU processor
TAU	Clock failure of SPU processor
DARY	Continuous ready (access to faulty memory)
KZU	Bracket counter overflow

Table 3-28 continued:

Abbrev.	Meaning
	Low word
KB	No block
KDB	No data block
STS	Soft stop
TLAF	Transfer/load error
SUF	Substitution error
STUEB	BSTACK overflow
STUEU	ISTACK overflow
NAU	Power failure
ZA	Timed interrupt (delayed interrupt, real time interrupt
QVZ	Timeout
ADF	Addressing error
PARE	Parity error
ZYK	Cycle time error
STOP	Mode selector switched to STOP
HOLD	DMA request from SPU processor

Operating Statuses and Program Execution Levels

4

Contents of Chapter 4

4.1 Introduction and Overview ..... 4 - 4

4.2 Program Execution Levels ..... 4 - 7

4.3 STOP Mode ..... 4 - 12

4.3.1 SOFT STOP ..... 4 - 12
4.3.2 HARD STOP ..... 4 - 16
4.3.3 OVERALL RESET ..... 4 - 17

4.4 START-UP Mode ..... 4 - 19

4.4.1 MANUAL and AUTOMATIC COLD RESTART ..... 4 - 21
4.4.2 MANUAL and AUTOMATIC COLD RESTART WITH MEMORY ..... 4 - 22
4.4.3 Comparison between COLD RESTART and COLD RESTART WITH MEMORY ..... 4 - 25
4.4.4 User Interfaces for Start-Up ..... 4 - 26
4.4.5 Interruptions during START-UP ..... 4 - 27
4.4.6 Examples of Programs for START-UP ..... 4 - 29

4.5 RUN Mode ..... 4 - 31

4.5.1 Cyclic Program Execution ..... 4 - 33
4.5.2 Specifying Time and Interrupt-Driven Program Execution ..... 4 - 36
4.5.3 Time-Controlled Program Execution ..... 4 - 37
Delayed interrupt ..... 4 - 38
Real time interrupt ..... 4 - 39
Cyclic timed interrupts ..... 4 - 41
4.5.4 Interrupt-Driven Program Execution ..... 4 - 45
Disabling interrupt-driven processing ..... 4 - 47
Reaction time ..... 4 - 48
Program execution levels and flags ..... 4 - 48

Operating Statuses and Program Execution Levels

4

This chapter provides an overview of the operating statuses and program execution levels of the CPU 948R. It informs you in detail about various types of start-up and the organization blocks associated with them, in which you can program your own sequences for various situations when restarting.

You will also learn the characteristics of the program execution modes "cyclic processing", "time-controlled processing" and "interrupt-driven processing" and will see which blocks are available for your user program.

4.1 Introduction and Overview

With the CPU 948R, there are four operating statuses, as follows:

- **START-UP** mode
- **RUN** mode
- **SOFT STOP** mode
- **HARD STOP** mode

In the START-UP and RUN modes, certain events can occur to which the system program must react. In many situations, an organization block (one from OB 1 to OB 37) is called as the user interface for the specific event.

The modes are displayed by LEDs on the front panel of the CPU. Some of the modes must be activated using the operating elements on the front panel of the CPU. The position of the LEDs and operating elements can be seen in Fig. 4-1.

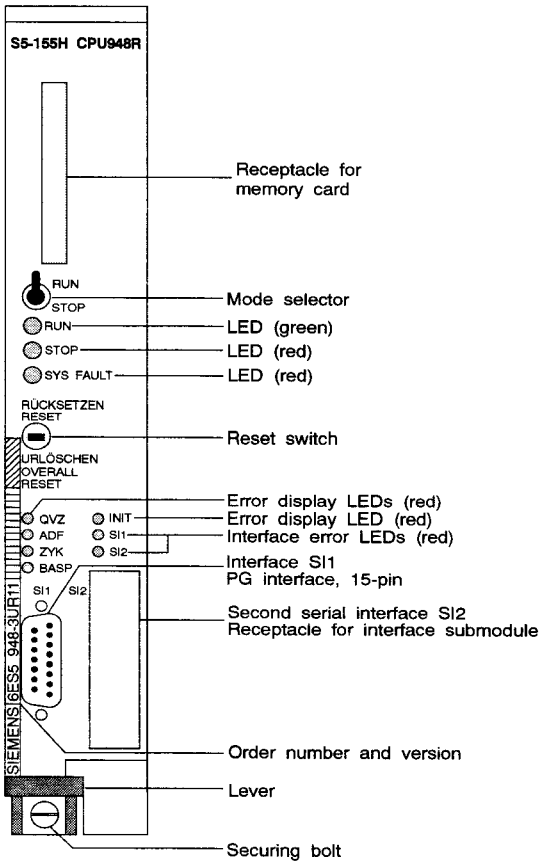


Fig. 4-1 Front panel of the CPU 948R with display and operating elements

Display of the modes by (LEDs)

Various LEDs on the front panel of the CPU indicate its current mode. The following table shows the relationship between the STOP and RUN LEDs and SYSFAULT LED and the corresponding mode. These displays are supplemented by further LEDs (BASP, ADF, QVZ, ZYK, INIT, SI1, SI2).

Table 4-1 Meaning of the LEDs "RUN" , "STOP" and "SYSFAULT"

LED			Meaning
"RUN"	"STOP"	"SYS-FAULT"	
RUN or START-UP mode:			
on	off	off	The CPU is in the RUN mode and is master (cyclic operation).
flashes	off	off	The CPU is in the RUN mode and is standby (cyclic operation).
on	flashes	off	The CPU is master, the parallel link has broken down.
on	on	on	Appears briefly after switching on.
on	on	off	The CPU is running through the start-up self-test.
off	off	off	The CPU is in the START-UP mode or the PG function "program test" is active.
SOFT STOP mode:			
off	on	off	Display when the power is switched on and the mode selector is set to STOP and no error occurred in the initialization. A START-UP is possible.
off	flashes quickly	off	Display when OVERALL RESET was requested by the operator or by the CPU itself. A START-UP is only possible if you go through an OVERALL RESET or remedy the error and then do the overall reset.
off	flashes slowly	off	This display occurs in the following situations: - An error has occurred in the RUN mode, but no appropriate error reaction has been programmed (exceptions: see Chapter 5). If you change the switch from RUN to STOP, the LED is once again lit permanently as long as the error does not occur again. - Operator error, e.g. selecting an invalid type of start-up, DB 1/DX 0 error etc. - When a stop statement is programmed (STP and STS). - With the PG function "end program test". - When a programming error or device fault occurs, (as an aid to locating the problem, the LEDs ADF, QVZ or ZYK light up with some of these errors).
on	flashes slowly	off	Defective parallel link.

LED			Meaning
"RUN"	"STOP"	"SYS-FAULT"	
Table 4.1 continued:			
HARD STOP mode:			
off	off	on	This display occurs in the following situations: - With a timeout (QVZ) or parity error (PARE) in the system RAM. - With ISTACK overflow. - When the STW STEP 5 operation is programmed.
"Coupling standby" mode:			
flashes	flashes	off	The CPU is the standby and the mode is "coupling standby".

Signalling and error LEDs

"BASP" LED — Command output is disabled, the digital outputs are switched to the safe state (to '0').

"QVZ" LED — Timeout of an I/O module (for more detailed information refer to Section 5.6).

"ADF" LED — Addressing error; the user program has accessed an address in the process image for which there was no module inserted in the I/Os at the last cold restart or which was not entered in DB1.

"ZYK" LED — Cycle time exceeded.

"INIT" LED — During initialization following POWER UP and with system errors, this LED is lit **constantly**.

"SI1" LED — on: no communication possible; internal error
off: interface SI1 is initialized and operational

"SI2" LED — This LED is always off; otherwise there is a CPU error

4.2 Program Execution Levels

Fig. 4-2 provides you with an overview of the program processing levels in the various modes. The explanations of the abbreviations are on the following page.

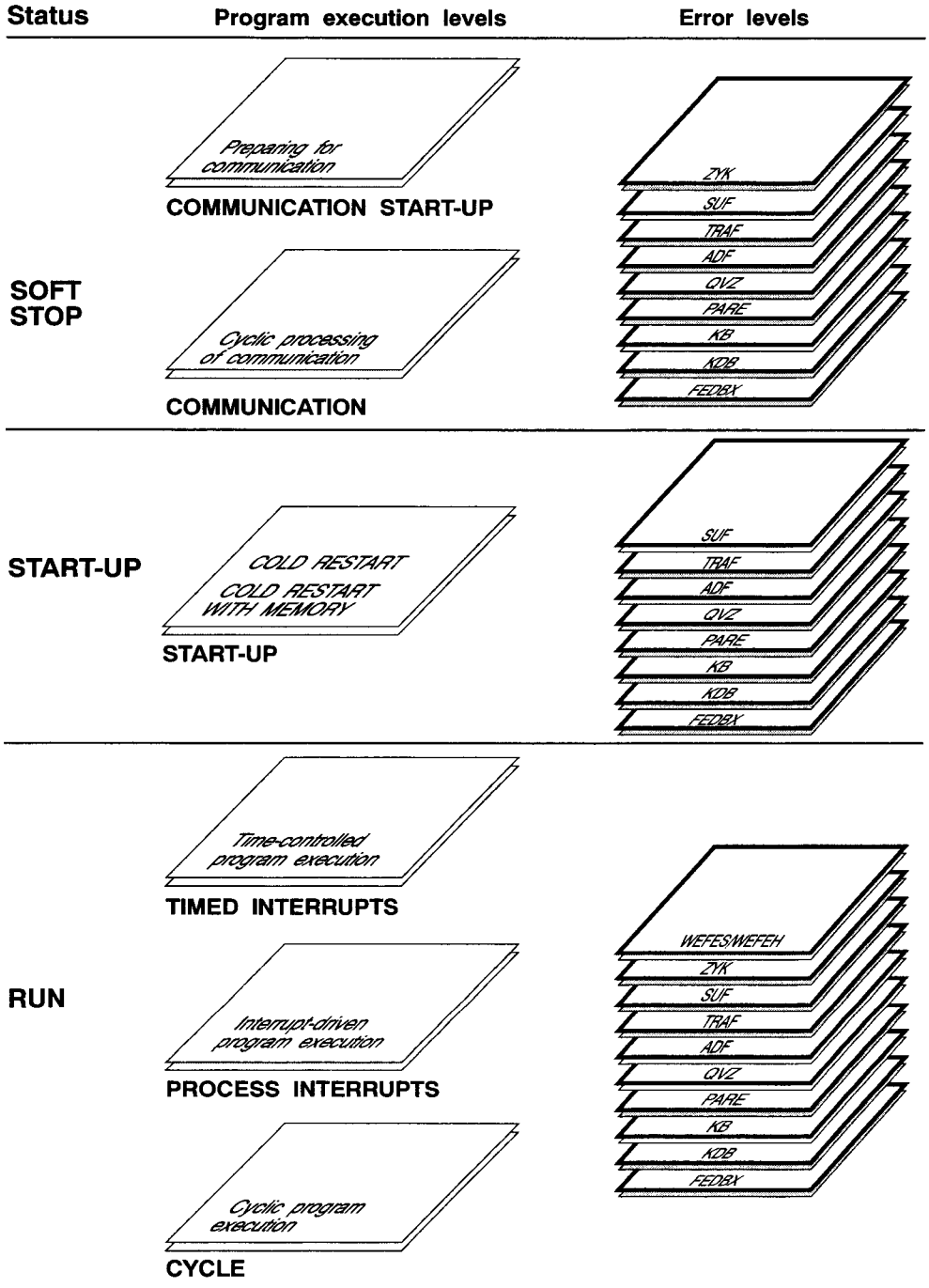


Fig. 4-2 Program execution levels

Table 4-2 Program execution levels

Level	Meaning	Priority
Error levels		
WEFES/WEFEH ZYK SUF TRAF ADF QVZ PARE KB KDB FEDBX	Collision of timed interrupts Cycle error Substitution error Transfer/load error Addressing error Timeout Parity error Called code block does not exist Called data block does not exist Error generating a data block, DB or DX	Each error handling routine has the highest priority. If an error occurs, the corresponding error level is nested in immediately.
Program execution levels in SOFT STOP		
COMMUNICATION START-UP COMMUNICATION	Preparation (start-up) for communication Cyclic processing of communication	–
Program processing levels in START-UP		
COLD RESTART COLD RESTART WITH MEMORY	Defined start of the user program Defined start of the user program, certain data remain as they were before the START-UP	–
Program execution levels in RUN		
TIMED INTERRUPTS PROCESS INTERRUPTS CYCLE	Time-controlled program execution Interrupt-driven program execution Cyclic program execution	Ascending priority (default) [1)]

1) The default can be changed by selecting parameters for DX 0 (refer to Chapter 7).

Processing via the system program

A specific system program is responsible for each level.

Interrupt stack (ISTACK)

If an interrupt occurs, the system program sets up an information field in the ISTACK for each level, to allow it to continue in the interrupted level after servicing the interrupt.

Nesting other levels

When an event occurs, which requires higher priority processing, the current level is interrupted by the system program and the higher priority level is activated.

This occurs in the following situations:

- at error levels: always at operation boundaries,
- all other levels: at block boundaries

Note
A maximum of 5 error organization blocks can be nested. If 5 error levels are activated simultaneously, this causes an ISTACK overflow and the CPU changes to the **HARD STOP**.

A specific **program execution level** is assigned to one or a group of organization blocks which are called by the system program after an event. If, for example, OB 9 is called to process a real time interrupt, the program execution level TIMED INTERRUPTS is activated.

After the system program calls an organization block, the CPU executes the STEP 5 statements is contains. The current register record is saved in the ISTACK and a **new register record** is set up (register: ACCU 1 to 4, block stack pointer, block address register, data block start address, data block length, step address counter, base address register and the interrupt condition code words ICMK and ICRW).
If "normal" program execution is interrupted by the occurrence of an event, following the execution of the OB, the CPU continues the program execution at the point of interruption (including all the blocks nested there) as long as no stop is programmed in the OB.

Sub-levels

The TIMED INTERRUPTS level contains several sub-levels to which a specific program (OB) is assigned. Within the TIMED INTERRUPTS level, the sub-levels have their own priority (refer to the following table).

TIMED INTERRUPTS level	
Sub-level	**Priority**
Delayed interrupt cyclic timed interrupt, shortest period ... ... cyclic timed interrupt, longest period real time interrupt	ascending priority (default)

Examples

Example of
"Execution by the system program":

At the CYCLE program execution level, the system program updates the process image of the inputs and outputs, triggers the cycle monitoring time and calls the PG interface management (system checkpoint).

Example of
"Interrupt stack":

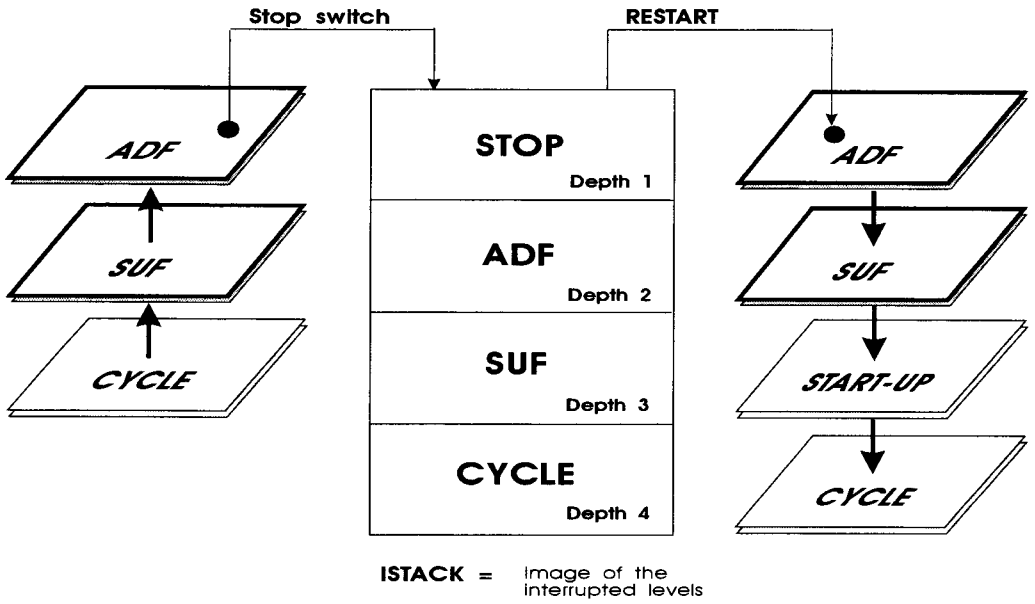


Fig. 4-3 Principle of changing level and the ISTACK

Example of
"Interrupting a basic level with interruptability at block boundaries":

A timed interrupt occurs while a process interrupt is being serviced. Since the timed interrupt has a higher priority, the servicing of the PROCESS INTERRUPT is interrupted at the next block boundary and the TIMED INTERRUPT level nested in. If an addressing error now occurs while servicing the timed interrrupt, the timed interrupt servicing is interrupted immediately at the next operation boundary to nest in the ADF level.

4.3 STOP Mode

The CPU 948R has two different STOP modes, the "hard" STOP and the "soft" STOP (= CPU capable of communication).

4.3.1 SOFT STOP

The SOFT STOP mode has the following features:

The CPU can communicate: the system program calls organization block OB 38 once after POWER UP (COMMUNICATION START-UP level in SOFT STOP) and then calls OB 39 (COMMUNICATION level in SOFT STOP).

Initialization of communication (OB 38)

To initialize communication, the system program calls **OB 38** as the user interface.

OB 38 is only called after **POWER UP** . The call is not dependent on the type of start-up (AUTOMATIC COLD RESTART/AUTOMATIC COLD RESTART WITH MEMORY) set in data block DX 0.

Start-up monitoring of OB 38

The time required for the execution of OB 38 is not monitored by the system program. You can, however, abort execution by changing the mode selector to STOP.

If an error occurs in OB 38, its execution is aborted and OB 39 is called if it exists.

Cyclic communication (OB 39)

If the cyclic program is interrupted causing a change to the SOFT STOP mode, **OB 39** is called as the user interface.

If the interruption of the cyclic program means that a handling block (communication) is not completely executed, it is possible and permitted to re-call the same block or the same block type in OB 39 (e.g. SEND).

Timer processing in OB 38/39

While OB 38/39 is running, timers and counters are not processed. If necessary, timers must then be processed with OB 121 or OB 150.

Monitoring the execution time of OB 39

The execution time of OB 39 is monitored by the system program. If execution takes longer than 2.55 seconds (fixed value), the system program detects a cycle time error; it then calls the error OB, OB 26, and then processes OB 39 again from the beginning. If a cycle time error occurs again, an ISTACK depth > 5 causes an ISTACK overflow (reaction: see following paragraph).

Reaction to an error in OB 39

If an error occurs in OB 39 or in a block called in OB 39 (e.g. QVZ), the system program calls the appropriate error organization block. After this has been executed, the program processing is continued in OB 39. If the error OB does not exist, OB 39 is executed again from the beginning. (Exception: with QVZ, KB and errors in the self-test there is no reaction).
If further errors occur, an ISTACK depth > 5 causes an ISTACK overflow:
the system program aborts program execution (OB 39 is no longer called), the CPU however remains in the SOFT STOP mode.

Data are not reset

If cyclic execution has already taken place in the RUN mode, the values of counters, timers, flags and the process image are retained during the transition to the stop mode.

Real-time clock

The real-time clock continues to run. It is updated in the RS area at 10 ms intervals.

BASP signal

The BASP (disable output command) signal is active. This disables all the digital outputs (exception: with the PG function "force outputs", BASP is not active).

ISTACK

If a user program was processed prior to the stop mode, information is entered in the interrupt stack (ISTACK) providing information about the cause of the interruption.

.

OB 38/OB 39 call

Figures 4-4 and 4-5 illustrate the principle of the OB 38 and OB 39 calls.

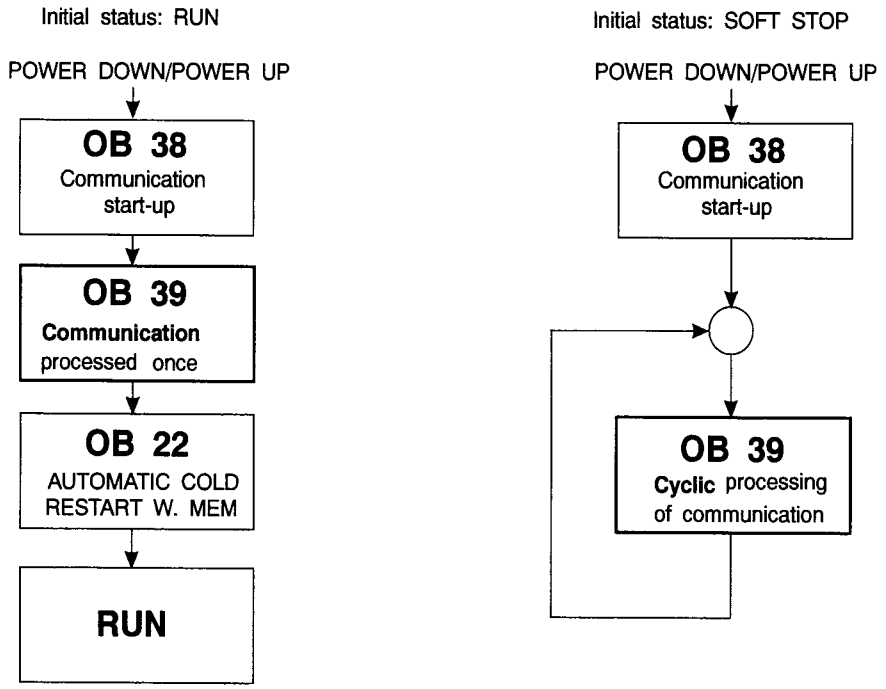


Fig. 4-4 Program execution after POWER UP

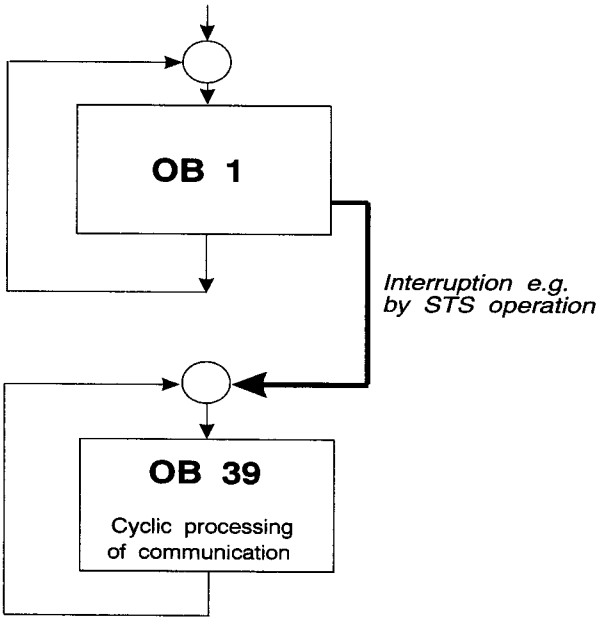


Fig. 4-5 Program execution after a cycle interruption

LED displays

The SOFT STOP status can be recognized by the LEDs on the front panel of the CPU as follows:

LED	Status
RUN	off
STOP	on (continuous or flashing light)
SYSFAULT	off
BASP	on (except with the PG function "force outputs")

The **STOP LED** signals the possible causes of the current stop status, as follows:

STOP LED lit continuously

The SOFT STOP mode was triggered by the following:

- by changing the mode selector from RUN to STOP,
- by the PG function PLC STOP,
- following an OVERALL RESET.

STOP LED flashes slowly (approx. once every 2 sec)

The SOFT STOP was triggered by the following:

- STP or STS statement in the user program,
- operator error (DB 1/DX 0 error, selection of an illegal start-up mode etc.),
- BSTACK overflow (STUEB) or bracket counter overflow (KZU),
- programming errors or device faults; the following LEDs provide further information:
  - "ADF" LED
  - "QVZ" LED
  - "ZYK" LED
- by the PG function "program test end" on this CPU.

Exiting the SOFT STOP status

The SOFT STOP status can be exited as follows:

a) by selecting a restart (refer to Section 4.4),

b) by an OVERALL RESET followed by a COLD RESTART.

4.3.2 HARD STOP

If the system program can no longer be executed properly, the CPU changes to the HARD STOP mode to ensure a safe mode in this situation.

A HARD STOP can be caused by the following:

- stop operation (STW) for the system program
- ISTACK overflow (STUEU),
- timeout (QVZ) or parity error (PARE) in the system RAM/EPROM,
- error in system program

> **Note**
> The CPU is stopped. You can only exit the HARD STOP mode by switching the power off and then on again!

LED displays

The HARD STOP status can be recognized by the following LEDs on the front panel of the CPU:

LED	Status
RUN	off
STOP	off
SYSFAULT	on
BASP	on

4.3.3 OVERALL RESET

With an OVERALL RESET the whole user memory and operand areas (flags, process images etc.) are deleted.

Requesting an OVERALL RESET

Before an OVERALL RESET can be performed, it must be requested. An OVERALL RESET is requested when the STOP LED flashes quickly.

Request by the system program

Each time you switch on the power, the CPU runs through an initialization routine. If an error is detected during the initialization (back-up voltage failure during POWER OFF), the system program requires an OVERALL RESET.

The cause of the problem must be eliminated (e.g. replacing the battery) following which an overall reset of the CPU is necessary.

An OVERALL RESET is also requested if a CPU or system error occurs. You can recognize this error because the request occurs again following the OVERALL RESET. In this case, contact your local Siemens representative.

Operator request

With the steps outlined in the table, you can also request an OVERALL RESET (the operating elements are on the front panel of the CPU - Fig. 4-1):

Step	Action	Result
1	Change the mode selector switch from RUN to STOP.	The CPU is in the STOP mode. The STOP LED is lit continuously.
2	Hold the mode switch in the OVERALL RESET position; at the same time, change the mode selector from STOP to RUN and then back to STOP.	An OVERALL RESET is requested. The STOP LED flashes quickly.

Note
If you do not want to execute the OVERALL RESET you requested, carry out a MANUAL COLD RESTART

Performing an OVERALL RESET

Regardless of whether you or the system program requested the OVERALL RESET, you perform the OVERALL RESET as follows (initial status: STOP LED flashing quickly):

OVERALL RESET using control elements on the CPU

Action	Result
Hold the reset switch in the OVERALL RESET position. At the same time, change the mode selector from STOP to RUN and back to STOP again (refer to Fig. 4-1).	An OVERALL RESET is performed. The STOP LED is lit continuously.

OVERALL RESET from the PG

Action	Result
Activate the PG function "delete all blocks".	An OVERALL RESET is performed. The STOP LED is lit continuously.

Note

If you do an OVERALL RESET on the CPU 948R from the PG, only the **connected** CPU is reset. In this case, you must change the CPU to the STOP mode before the overall reset. The other CPU remains in the RUN mode.
If the connected CPU is **not** in the STOP mode, the error message "Memory error in PLC" appears on the PG screen.

During the OVERALL RESET, all the LEDs are off apart from the INIT and BASP LEDs. During the overall reset self-test, the RUN and STO LEDs are lit continuously.

Once the OVERALL RESET is completed, the only permitted start-up mode is then a COLD RESTART.

Loading the memory card and calling OB 39

After performing the OVERALL RESET, the memory card is loaded (provided it is plugged in) following which OB 39 is called.

4.4 START-UP Mode

> **Note**
> The information about the START-UP applies to both CPUs. The standby CPU, however, takes all the status data from the master CPU before entering the CYCLE.

The START-UP mode has the following features:

Mode change

START-UP is the transition from the STOP mode to the RUN mode.

Start-up types

The CPU 948R has the following start-up modes:

- COLD RESTART (manual or automatic)
- COLD RESTART WITH MEMORY (manual or automatic)

(You can select the type of start-up with the operating elements and by assigning parameters in DX 0)

COLD RESTART

The cyclic user program is processed from the beginning.

The system program calls **OB 20** as the user interface.

COLD RESTART WITH MEMORY

The same as COLD RESTART, but flags, timers, counters and process images are not deleted.

The system program calls the following OBs as the user interface:

- **OB 21** for a MANUAL COLD RESTART WITH MEMORY,
- **OB 22** for an AUTOMATIC COLD RESTART WITH MEMORY.

No time monitoring of the start-up OBs

The execution time of the start-up organization blocks is **not** monitored. You can call other blocks within these OBs. Make sure you avoid endless loops!

BASP

The BASP signal (disable output command) is active. This disables all the digital outputs:

Interrupts

The interrupts (normal interrupts, process interrupts, timed interrupts) are disabled.

Start-up self-test

With all types of start-up, the system program runs through a complete self-test. If an error is detected, the CPU of the subunit affected changes to the SOFT STOP and outputs an error message. During the start-up self-test, both the RUN and STOP LEDs are lit continuously.

Subunit assignment

During a COLD RESTART, the system program checks whether I/Os are plugged in which are specifically assigned to subunit A or B (one-sided I/Os). If this is the case, the assignment subunit A - subunit B is derived from this. If there are no such I/Os, subunit A is the subunit in which the IM 324R is plugged in. If the IM 324R module is defective, this becomes subunit B.

If no one-sided I/Os are plugged in, the subunit assignment is only re-determined after the next OVERALL RESET.

Master-standby assignment

If the self-test is completed successfully, the subunit signals that it is operational via the parallel link IM 324R. The system program then checks whether the partner PLC is already operational. If it is, the subunit becomes standby. If the partner PLC is not yet operational, the subunit becomes master.
When there is a cold restart on the master, the passivation is cancelled. The error data block is completely deleted.

Start-up phase of the master CPU

So that serious software and hardware configuration errors can be detected and indicated before the standby is coupled, the COM 155H configuration must be compared with the actual hardware configuration. If, for example, the I/Os were configured as two-channel and is plugged in as switching, the master CPU has access to its I/Os configured as two-channel. It must, however, establish whether or not it would have access as standby CPU. If the master CPU recognizes that the I/Os configured as two-channel are plugged in as switching, it signals the error and switches the I/Os passive

During START-UP, the system program of the master CPU sets the IM 314R to the "Test" mode and then immediately to "bus access". The "test and bus access" mode is retained until the complete start-up OB (OB 20, OB 21 or OB 22) has been processed. The system program then switches off the test mode on the IM 314R.

START-UP without standby

If the standby is not operational due to an error in the START-UP, the master PLC starts without a standby. This mode is the same as the solo mode.

Generating the process image list during START-UP

During START-UP, the process image list for digital inputs and outputs is created according to the data configured with COM 155H, regardless of which modules are actually plugged in.

4.4.1 MANUAL and AUTOMATIC COLD RESTART

When is a COLD RESTART permitted?

A COLD RESTART is **always permitted** provided the system is not requesting an OVERALL RESET.

When is a COLD RESTART necessary?

A COLD RESTART is **necessary** after the following:

- OVERALL RESET,
- loading the user memory with the user program while the CPU is in the stop mode
- ISTACK/BSTACK overflow,
- COLD RESTART aborted (by POWER OFF or changing the mode selector to "STOP"),
- stop after PG function "program test end"
  inconsistent data are detected in the user memory after a COLD RESTART WITH MEMORY

MANUAL COLD RESTART

You can **trigger** a MANUAL COLD RESTART as follows:

- Using the operating elements of the CPU:

  Hold the reset switch in the RESET position; at the same time change the mode selector from STOP to RUN (refer to Fig. 4-1).

- From the PG:

  Select the PG function PLC START/COLD RESTART.

AUTOMATIC COLD RESTART

An AUTOMATIC COLD RESTART is **triggered** as follows:

At POWER UP, when

- "AUTOMATIC COLD RESTART after POWER UP" is set in DX 0 **and** the CPU was not in the STOP mode when the power was switched off,

> **Note**
> If the CPU was in the STOP mode when the power was switched off (e.g. after an addressing error), then **AUTOMATIC** COLD RESTART is **not permitted**. The STOP mode can then only be exited with a **MANUAL** COLD RESTART.

Aborting a COLD RESTART

You can abort an active COLD RESTART only by changing the mode selector to STOP or by switching off the power. If you abort a COLD RESTART you must repeat it.

4.4.2 MANUAL and AUTOMATIC COLD RESTART WITH MEMORY

When is a COLD RESTART WITH MEMORY possible and permitted?

A MANUAL COLD RESTART WITH MEMORY is **only possible** after stoppages caused by the following:

- the mode selector was changed from the RUN position to the STOP position,
- POWER OFF, with the appropriate setting in data block DX 0,
- PG function PLC STOP.

Note
If the stoppage was caused by an event other than those listed above, then **no COLD RESTART WITH MEMORY** is possible. The system program will only permit a COLD RESTART (without memory).

A MANUAL or AUTOMATIC COLD RESTART WITH MEMORY is **only permitted** when

- the user program was not modified during the stop mode

  and

- a COLD RESTART (without memory) is not necessary for other reasons (refer to Section 4.4.1).

MANUAL COLD RESTART WITH MEMORY

You **trigger** a MANUAL COLD RESTART WITH MEMORY as follows:

- using the control elements of the CPU:

  initial state: the reset switch is in the mid setting

  change the mode selector from STOP to RUN (refer to Fig. 4-1)

- from the PG:

  select the PG function PLC START/WARM RESTART.

AUTOMATIC COLD RESTART WITH MEMORY

An AUTOMATIC COLD RESTART WITH MEMORY **is triggered** as follows:

With POWER UP, when

- the default "AUTOMATIC COLD RESTART WITH MEMORY after POWER UP" is set in data block DX 0 or DX 0 does not exist,

- the mode selector on the CPU is set to RUN
  and
  the CPU was not in the STOP mode when the power was switched off,

- no further errors have occurred during the initialization nor before the power was switched off,

- no COLD RESTART (without memory) is required due to the reasons listed above.

Following power failure or switching the power off in the RUN mode followed by the return of power/POWER UP, the CPU runs through an initialization routine and then automatically performs a COLD RESTART WITH MEMORY.

When the power returns in a central controller and there is a power failure on an expansion unit, (PEU signal), the CPU remains in the STOP mode. It remains in this mode until the PEU signal is switched inactive (however only for a maximum of 12 seconds) and then performs an AUTOMATIC COLD RESTART WITH MEMORY or an AUTOMATIC COLD RESTART. The subunit that starts up as standby goes through a COLD RESTART.

Note
With a COLD RESTART WITH MEMORY note the following special situations:

- STOP during error processing:
  The CPU is currently processing an error OB (e.g. due to an addressing error ADF) and then changes to the STOP mode owing to POWER OFF, HALT, stop switch or PG-STP. Following this, a MANUAL or AUTOMATIC COLD RESTART WITH MEMORY is executed.

  Reaction of the CPU:

  - **Before** OB 21/22 is called, the interrupted processing of the error OB is completed.
  - If the error OB does not lead to a stop operation, then following the processing of the remainder of the error OB, a **COLD RESTART WITH MEMORY** is executed.
  - If the error OB sets the CPU to the stop mode, then only a **COLD RESTART** (without memory) is possible.

- Both subunits change to STOP due to POWER OFF, HALT, stop switch or PG-STP:

  Reaction of the PLCs

  If POWER UP occurs simultaneously for both subunits (difference <50 ms), the system program makes sure that the previous master (with up-to-date process data) becomes master again. If, for any reason, the previous master does not restart, the previous standby becomes master and enters a message in the error DB.

Aborting a COLD RESTART WITH MEMORY

You can only abort a COLD RESTART WITH MEMORY after it has started by changing the mode selector to STOP or by POWER OFF. If you abort the warm restart in this way, both a COLD RESTART or COLD RESTART WITH MEMORY is then possible.

4.4.3 Comparison between COLD RESTART and COLD RESTART WITH MEMORY

The following table contains a comparison of the start-up types COLD RESTART and COLD RESTART WITH MEMORY.

Table 4-3 Differences between COLD RESTART and COLD RESTART WITH MEMORY

System program function	COLD RESTART	COLD RESTART WITH MEMORY
Evaluation of:		
- DB 1	yes	no
- DX 0	yes	no
- DX 1 (create address lists for updating process images)	yes	no
Create block address list in DB 0	yes	no
Delete:		
- ISTACK/BSTACK	yes	yes
- Process image of inputs	yes	no
- Process image of outputs	yes	no
- Digital outputs (2 * 128 bytes)	yes	no
- Analog outputs (2 * 128 bytes)	yes	no
- IPC flags	yes	no
- F and S flags	yes	no
- Timers and counters	yes	no
- Error DB	yes	no
Cancel passive	yes	no
Start-up self-test	yes	yes
Display of restart type on PG in ISTACK control bits	NEUDF	WIEDF

4.4.4 User Interfaces for Start-Up

The organization blocks OB 20, OB 21 and OB 22 serve as user interfaces for the various types of start-up. You can store your STEP 5 program for the type of start-up in these blocks.

OB 20

When the CPU executes a MANUAL or AUTOMATIC COLD RESTART, the system program calls OB 20 **once**. In OB 20, you can store a STEP 5 program which is responsible for preliminary steps for a **cold restart** of cyclic processing prior to the execution of the cyclic program.

You can, for example:

- set flags
- start timers (the start is executed by the system program when it enters the RUN mode)
- set default values for data to be output to I/O modules
- synchronize CPs.

After processing OB 20, the cyclic program begins by calling OB 1. If OB 20 is not loaded, the CPU begins the cyclic program execution immediately after the COLD RESTART is completed (following the system activities).

OB 21

If the CPU performs a MANUAL COLD RESTART WITH MEMORY, the system program calls OB 21 once if it is loaded. Here, you can store a STEP 5 program which executes preliminary steps for **resuming** the cyclic program.

MANUAL COLD RESTART WITH MEMORY

After processing OB 21, the system program executes a **COLD RESTART WITH MEMORY**, (the CPU continues program execution with the **first STEP 5 statement in OB 1**). The signal states of the flags, IPC flags and the block address list (DB 0) **are retained**.

OB 22

When the CPU executes an AUTOMATIC COLD RESTART WITH MEMORY, the system program calls OB 22 once if it is loaded. Here, you can store a STEP 5 program which executes preliminary steps (generally following a power failure) for **resuming** cyclic program execution.

AUTOMATIC COLD RESTART WITH MEMORY

After processing OB 22, the system program executes a **COLD RESTART WITH MEMORY**, (the CPU continues program execution with the **first STEP 5 statement in OB 1**). The signal states of the flags, IPC flags and the block address list (DB 0) **are retained.**

4.4.5 Interruptions during START-UP

A start-up program can be interrupted by the following:

- Power failure in the central controller (NAU) or in the expansion unit (PEU),
- Stop switch, stop command, HALT or PG-STP

  or

- Program errors or device faults (refer to Section 5.7).

Basic rules for an interrupted START-UP

The following basic rules apply to the start-up response of the CPU 948R:

If the START-UP is interrupted, the subsequent START-UP is always restarted **from the beginning.**
The last selected type of start-up is selected. Example: 1. POWER DOWN (NAU) in the cycle 2. Switch set to STOP 3. POWER UP 4. Switch set to RUN Reaction: the CPU executes a MANUAL COLD RESTART WITH MEMORY
An interrupted COLD RESTART cannot be continued with a COLD RESTART WITH MEMORY, but must be repeated.
Following an interrupted COLD RESTART WITH MEMORY, both a COLD RESTART and a new COLD RESTART WITH MEMORY are possible.

Response of the CPU on the return of power after power failure

If the start-up is interrupted by a power failure, the response of the CPU when power returns depends on the **set** and **interrupted** mode. The following table provides an overview.

Mode set in DX 0	Interrupted mode	Interrupted start-up OB	Reaction of the CPU
AUTOMATIC COLD RESTART	MANUAL COLD RESTART	OB 20	Depending on the prior events, an **AUTOMATIC COLD RESTART** is executed (OB 20 is called and processed from the beginning). No "**remaining start-up**" is processed, (OB 20 or OB 21 is not resumed).
	MANUAL COLD RESTART WITH MEMORY	OB 21	
	AUTOMATIC COLD RESTART	OB 20	
AUTOMATIC COLD RESTART WITH MEMORY	MANUAL COLD RESTART	OB 20	STOP
	MANUAL COLD RESTART WITH MEMORY	OB 21	An AUTOMATIC COLD RESTART WITH MEMORY is executed (OB 22 is called and processed from the beginning); no "**remaining start-up**" is processed (OB 21 or OB 22 is not resumed).
	AUTOMATIC COLD RESTART WITH MEMORY	OB 22	

4.4.6 Examples of Programs for START-UP

This section contains simple examples of programs for H system applications, The sequences of operations can be programmed in the start-up OBs.

Direct access to I/Os

Switched I/Os must **only** be accessed directly by **the master** and must therefore be programmed in the start-up OBs OB 20, OB 21 and OB 22. Evaluating the H flag word (status byte) in the STEP 5 program shows whether the subunit is currently master or standby. This decides whether or not I/Os are accessed.

1. **Access to switched I/Os:**

```
          :
          :AN   F x.2
          :JC   =M001
          :T    PY 30       Access to switched I/O
M001      :
          :
```

2. **Access to *one-sided I/Os* (subunit A):**

```
          :
          :AN   F x.4
          :JC   =M001
          :T    PY 34       Access to subunit A I/O
M001      :
          :
```

3. **Access to *one-sided I/Os* (subunit B):**

```
          :
          :A    F x.4
          :JC   =M001
          :T    PY 36       Access to subunit B I/O
M001      :
          :
```

4. **Access to *redundant I/Os*:**

```
No scan necessary here!
```

Generating/deleting/ copying/duplicating blocks

Blocks can be generated or deleted using the operations G DB/GX DX or with the special functions OB 124, OB 125, OB 254 or OB 255 **only during START-UP and only by the master.** The example below shows how a sequence of block operations is skipped if the CPU is not the master. If the application requires such a sequence, it must be programmed in the start-up organization blocks.

```
Generate, delete blocks etc. only when the CPU is master:

          :
          :AN   F x.2      Not master?
          :JC   =RESE
          :
          :               Include operations
          :               to generate, delete
          :               and copy/duplicate
          :               blocks here
  RESE    :
          :
```

4.5 RUN Mode

When the CPU has executed a START-UP (and only then) it changes to the **RUN** mode. This mode is characterized by the following features:

Execution of the user program

The user program in OB 1 is executed cyclically and additional interrupt-driven program sections can be nested in it.

When executing the user program on the S5-155H, apart from the normal cyclic processing of the process images in the CPUs of the subunits, other system functions are also executed for synchronization, exchange and comparison of data and, in some situations. also operating modes.

Event-synchronized user program execution

The same user programs are run in the two subunits (master and standby) of the S5-155H. The master and standby operate **event-synchronized**. The 155H system program ensures that both subunits operate with the same data.

Timers, counters, process image

All the timers and counters started in the program continue to run.

Process image of the inputs (PII)

At the start of every cycle, the master and standby read the process images of the one-sided and redundant inputs assigned to them. The process image of the switched inputs is only read by the master. Both subunits then exchange the complete process image of the inputs.

BASP

The BASP signal (disable command output) is switched inactive. This enables all the digital outputs.

IPC flags

The IPC flags (if programmed in DB 1) are updated cyclically.

In the RUN mode, the following program execution levels exist:

CYCLE

The user program in OB 1 is processed cyclically.

PROCESS INTERRUPTS

The execution of the user program is interrupt-driven (1 process interrupt level with 8 sub-levels).

TIMED INTERRUPTS

The user program is processed time-controlled (9 cyclic timed interrupts, 1 delayed interrupt, 1 real time interrupt).

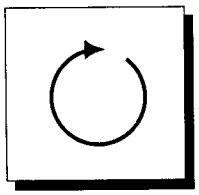
The execution levels differ from each other as follows:

- they are triggered by different events
- there are one or more organization blocks serving as user interfaces for each level of program execution.

All the processing levels in a CPU 948R can be programmed simultaneously. The levels are called by the system program according to the preset priority (refer to Section 4.2).

4.5.1 Cyclic Program Execution

With programmable controllers, **cyclic program execution** (program execution level **CYCLE**) is the main mode.



Triggering

If the CPU has completed the start-up program without errors, it then begins cyclic program execution.

Principle

The following figure illustrates the principle of cyclic program execution particularly the interaction between the subunits of the S5-155H.

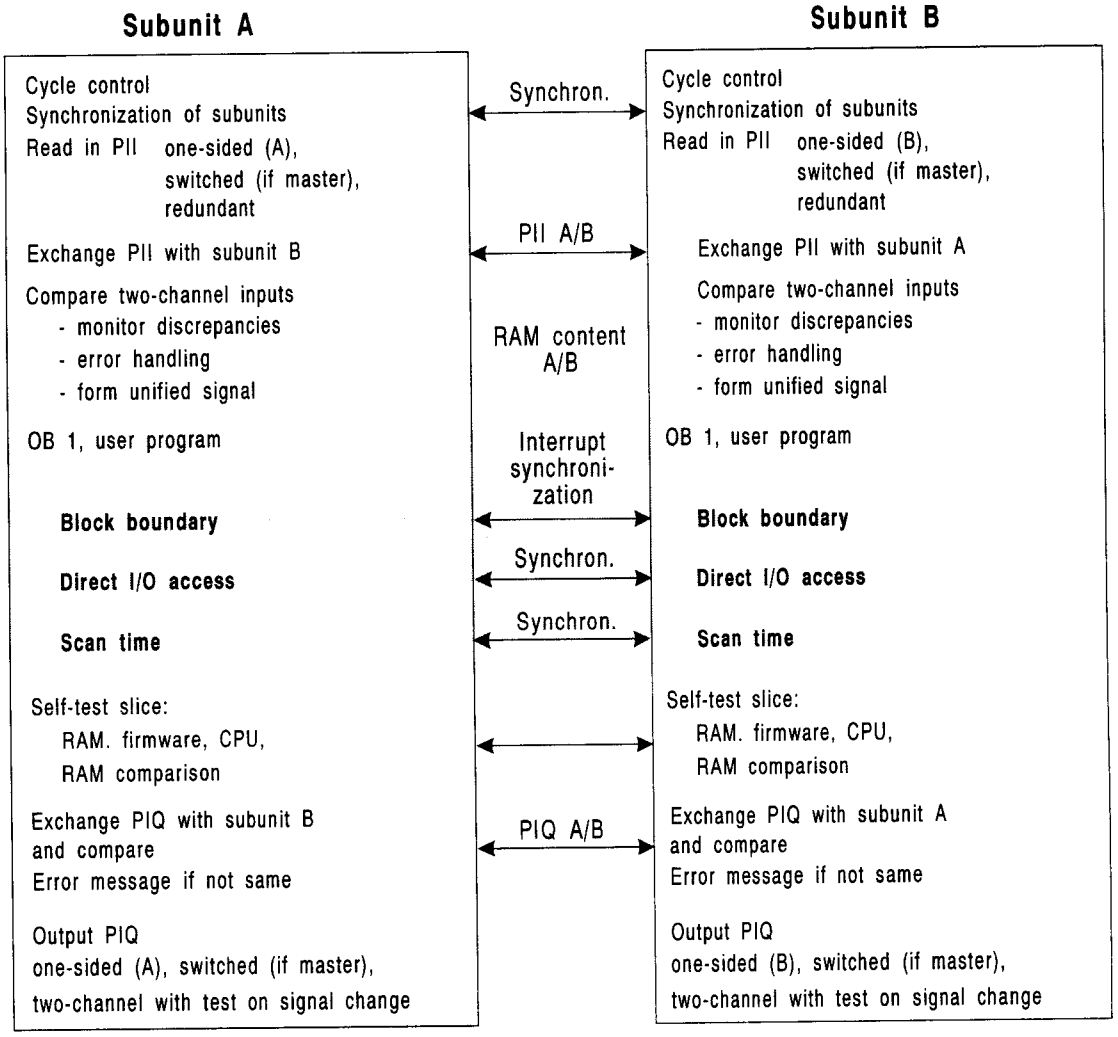


Fig. 4-6 Cyclic program execution with the S5-155H

User interface OB 1

During cyclic program execution, organization block OB 1 is called regularly as the user interface. The STEP 5 user program in OB 1 is processed from the beginning with the block calls you have programmed. After the system activities, the CPU starts again from the beginning with the first STEP 5 statement in OB 1.

In OB 1, you program the calls for program, function and sequence blocks to be executed within the cyclic program.

Interrupt points

Cyclic program execution can be interrupted at **block boundaries** by the following:

- time-controlled program execution
- process interrupt-driven program execution via input byte IB 0

Cyclic program execution can be interrupted or **aborted** as follows:

- when a device fault or program error occurs (at operation boundaries,
- by operator intervention:
  - stop switch, (at operation boundaries),
  - PG function (at checkpoints - refer to Chapter 10),
- by the stop command STS (at operation boundaries),
- by a power failure on the central controller (at operation boundaries).

Special situation: operation without connected EU

If the IM 314R module is not plugged in (no switched EU) or if it is faulty and the IM 324R of subunit A is also faulty, both subunits change to the RUN mode and become master.
The green RUN LEDs are lit continuously. The red STOP LEDs flash slowly.

Since this mode is not permitted, if you are operating an S5-155H without an IM 314R, you must set one of the subunits to STOP. If a bus-switchable EU with IM 314R is configured and exists, the system program 155H detects this illegal mode and changes the standby PLC to STOP.

Example of a program


```
No bus-switchable expansion unit (EU)is configured or exists.

-  A one-sided DQ (Q y.z) must be connected to one-sided DI (I y.z).

-  Function block FB nnn is called in OB 1.



FB nnn
SEGMENT 1          0000
NAME:CONNECT

0005        :A    F x.2                    am master and operating alone
0006        :                              (evaluation of H flag word)
0007        :=    Q x.y                    one-sided DQ in subunit A
0008        :A    F x.2
0009        :A    I x.y                    one-sided DI in subunit B
000A        :JC   =STS
000B        :BEU
000C  STS   :STS
000D        :BE

The LEDs on the front panel provide information about the mode of the
subunit:

•  Subunit is "master", the parallel link is intact
   -  RUN LED:    on
   -  STOP LED:   off

•  Subunit is "master", the parallel link is defective
   -  RUN LED:    on
   -  STOP LED:   flashes slowly

•  Subunit is "standby" in the redundant mode
   -  RUN LED:    flashes slowly
   -  STOP LED:   off
```

4.5.2 Specifying Time and Interrupt-Driven Program Execution

With time and interrupt-driven program execution, various types are available which can at present only be used as alternatives (i.e. not mixed). You decide which of the types of processing you want to use by setting the parameter in data block DX 0 (refer to Chapter 7).

"Process interrupts via IB 0"

The execution mode "process interrupts via IB 0" has the following features:

- the delayed interrupt (processed by OB 6) **cannot** be used,
- the real time interrupt (processing by OB 9) **cannot** be used.

"System interrupts" mode

The "system interrupts" mode is characterized by the following features:

- delayed interrupts are processed by OB 6,
- real time interrupts are processed by OB 9.

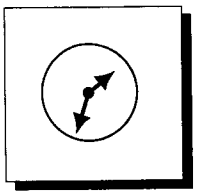
Interrupting time and interrupt-driven program execution

Time and interrupt-driven program execution can be interrupted or **aborted** as follows:

- when a device fault or program error occurs (at operation boundaries,
- by operator intervention:
  - stop switch, (at operation boundaries),
  - PG function (at checkpoints - refer to Chapter 10),
- by the stop command STS (at operation boundaries),
- by a power failure on the central controller or in the expansion unit (at operation boundaries).

4.5.3 Time-Controlled Program Execution

This type of program execution includes the delayed interrupt, the real time interrupt and cyclic timed interrupts.



All these interrupts are **time-controlled**.

Time-controlled program execution uses the **TIMED INTERRUPTS** level.

Delayed interrupts

Triggered once after a selected delay time in the millisecond range. Organization block OB 6 is called with this interrupt.

Real time interrupt

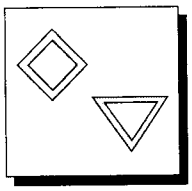
Triggered at a selected interval or once at an absolute point in time. Organization block OB 9 is called with this interrupt.

Cyclic timed interrupt

Triggered at 9 different intervals. Each timed interrupt is assigned an organization block (OB 10 to OB 18). This involves fixed cycles, i.e. the interval between two program stops is fixed.

Priorities

Within time-controlled program execution, the following priorities are set:



Delayed interrupt OB 6	OB 6	
cyclic timed int., period 1	OB 10, shortest period	
cyclic timed int., period 2	OB 11	
cyclic timed int., period 3	OB 12	
cyclic timed int., period 4	OB 13	ascending
cyclic timed int., period 5	OB 14	priority
cyclic timed int., period 6	OB 15	
cyclic timed int., period 7	OB 16	
cyclic timed int., period 8	OB 17	
cyclic timed int., period 9	OB 18, longest period	
real time interrupt	OB 9	

Note
Real time interrupt servicing in OB 6 and in OB 9 is only possible when the parameter "process interrupts via IB 0 = off" is set in DX 0. With the default setting in DX 0 ("process interrupts via IB 0 = on") the corresponding process interrupts of IB 0 are processed using OB 6 and OB 9 (refer to Section 4.5.4).

Delayed interrupt

With the delayed interrupt of the CPU 948R, small time intervals with a resolution of 5 ms can be set. Once the selected time has elapsed, the system program calls OB 6 **once**.

Resolution

The delayed interrupt is generated by calling the special function organization block OB 153 (refer to Section 6.13). As soon as the delay time assigned with OB 153 has elapsed, the system program interrupts the current program execution and calls OB 6 (program execution level TIMED INTERRUPTS). Following this, program execution is resumed at the point at which it was interrupted.

In the CPU 948R, OB 6 can only be called at block boundaries. This means that **the time of the longest running block** may be added to the selected delay time.

The use of the delayed interrupt is, however, only possible when "process interrupts via IB 0 = off" is set in the data block DX 0.

User interface OB 6

OB 6 is called as the user interface for a delayed interrupt. In OB 6, you write a STEP 5 program to be executed in this situation. If OB 6 is not loaded, program execution is not interrupted.

Interruptions

With the default setting, the TIMED INTERRUPTS level has the highest priority of the basic levels (can be modified by changing the parameter assignment in DX 0).

In timed-controlled program execution, the servicing of the delayed interrupt has highest priority.

Owing to the distribution of priorities, the processing of the delayed interrupt cannot be interrupted by any other user program.

Special features

- A delayed interrupt is only processed in the RUN mode. Delayed interrupts occurring in the STOP mode, during **power down** or START-UP are **discarded**.
- A generated delayed alarm (= OB 153 call was processed) is **not** retained in the transition to the STOP mode and during POWER OFF.

- If you generate a new delayed interrupt, i.e. call OB 153 with new parameters, a previously set delayed interrupt is cancelled. A delayed interrupt currently being processed is continued. This means that only **one** delayed interrupt is valid at any one time.
- If a delayed interrupt occurs without the previous one being completely processed, the new interrupt is discarded. **Delayed interrupts are not checked for collisions!**
- Note the special functions OB 122 and OB 142 with which you can disable or delay the servicing of delayed interrupts.

Real time interrupt

The CPU 948R has a battery-backed clock (central back-up via the power supply of the central controller), which you can set and read out using the STEP 5 program. This clock allows time-controlled execution of a program section.

While the delayed interrupt is used for fast events, real time interrupts are particularly suitable for processing events which occur **once** or which occur at **longer intervals**, e.g. hourly, daily or monthly. Once the point in time is reached, the system program calls OB 9.

Triggering

A real time interrupt (timed job) is generated by calling the special function organization block OB 151 (refer to Section 6.13). Once the time set in OB 151 is reached (a time, a date) the timed job is executed. The system program interrupts the current program execution and calls OB 9 (program execution level TIMED INTERRUPTS). Following this, program execution is resumed at the point at which it was interrupted.

To use the real time interrupt, "process interrupts via IB 0 = off" must be set in data block DX 0.

User interface OB 9

For a real time interrupt, OB 9 is called as the user interface. In OB 9, you write a STEP 5 program to be processed when the OB is called. If OB 9 is not loaded, program execution is not interrupted.

Interruptions

As default, the TIMED INTERRUPTS layer has the highest priority of the basic layers (can be modified in DX 0).

In time-controlled program execution, the execution of real time interrupts has the lowest priority. This can therefore be interrupted by the processing of a delayed interrupt or a cyclic timed interrupt.

Special features

- A real time interrupt is only processed in the RUN mode. real time interrupts occurring in the STOP mode, when there is a **power failure** or during START-UP are **discarded.**
- Once a real time interrupt has been generated (= OB 151 call has been processed) it is retained in a WARM RESTART and following POWER DOWN/POWER UP; it is, however, deleted during a COLD RESTART.
- If you generate a new real time interrupt, i.e. call OB 151 with new time values, a previously set real time interrupt is cancelled. A real time interrupt currently being processed is continued. This means that only **one** real time interrupt is valid at any one time.
- If a real time interrupt occurs without the previous one being completely processed, the new real time interrupt is discarded. **Real time interrupts are not checked for collisions!**
- Note the special functions OB 122 and OB 142 with which you can disable or delay the servicing of real time interrupts.

Cyclic timed interrupts

On the CPU 948R, you can process 9 different time-controlled programs, each being called at a different cyclic interval.

Triggering

The basic clock pulse for timed interrupt processing is set to 100 ms. Using a special parameter in data block DX 0, you can adjust this in steps of 10 ms (basic clock pulse = yy * 10 ms where: 01H ≤ yy ≤ FFH).

You can base the setting of the clock pulse on the shortest time required by your application for cyclic processing.

Time interval

Cyclic timed interrupts are processed at fixed intervals with 9 possible intervals (periods). Each interval is assigned to a specific organization block. You can select between two sets of intervals. You select the sets of intervals using a special parameter in data block DX 0. the following table illustrates the two sets of intervals with the assignment of the different intervals to organization blocks.

Table 4-4 Sets of intervals and intervals of the TIMED INTERRUPTS

Interval set 1 (default)	**Interval set 2**	**OB called**
1x basic clock pulse	1 x basic clock pulse	OB 10
2 x basic clock pulse	2 x basic clock pulse	OB 11
5 x basic clock pulse	4 x basic clock pulse	OB 12
10 x basic clock pulse	8 x basic clock pulse	OB 13
20 x basic clock pulse	16 x basic clock pulse	OB 14
50 x basic clock pulse	32 x basic clock pulse	OB 15
100 x basic clock pulse	64 x basic clock pulse	OB 16
200 x basic clock pulse	128 x basic clock pulse	OB 17
500 x basic clock pulse	256 x basic clock pulse	OB 18

Note
The **first** TIMED INTERRUPT OB call following the start-up takes place within the time assigned to the OB.

If, for example, the interrupt time "500 s" is set for OB 18 (basic clock pulse setting in DX 0 = 1 s and time base = 1), then the first OB 18 call takes place after approximately 20 s following a COLD RESTART or COLD RESTART WITH MEMORY. All further calls are then at intervals of 500 s.

User interfaces OB 10 to OB 18

When a timed interrupt occurs, the corresponding organization block is called as the user interface at the next block boundary (or operation boundary).

For example, you would program the routine to be inserted in cyclic program execution every 100 ms in OB 10 (default).

The timed interrupt is only processed if the **assigned organization block is loaded**. If none of the organization blocks OB 10 to OB 18 are loaded, there is no time-controlled program execution and the cyclic program is not interrupted.

You can disable the execution of timed interrupts by setting a parameter in data block DX 0, e.g. for testing your program.

Interruptions

As default, the TIMED INTERRUPTS level has the highest priority of the basic levels (can be modified in DX 0).

Owing to the distribution of priority within time-controlled program execution, the following interruptions in the processing of a cyclic timed interrupt are possible:

- the processing of a cyclic timed interrupt can be interrupted by the processing of a delayed interrupt
- organization blocks with shorter time bases have higher priority and can interrupt organization blocks with longer time bases (e.g. OB 12 can interrupt OB 17).

Note
With the **three shortest time bases (OB 10 to 12)** multiple processing without interruption is possible. If, for example, while OB 10 is being processed, a further timed interrupt for OB 10 occurs, the currently active processing of OB 10 is first completed. Following this, OB 10 is called immediately again. If, however, there are more than **three timed interrupts** pending for one of the time bases, a **collision of timed interrupts** error occurs.

Collision of timed interrupts

In the CPU 948R there are two different types of collisions of timed interrupts:

Type of error/cause	ISTACK ID	Reaction of the CPU
Timed interrupt queue overflow: - there are more than three timed interrupts pending for one of the three shortest time bases (OB 10 to 12), - one of the other OBs (OB 13 to 18) is called again before it has been completely processed.	In the "ISTACK output" of the programmer, the error ID WEFES is marked in the control bits.	The system program calls OB 33 as the user interface. If this is not loaded, the CPU changes to the **stop mode.**
Due to the interrupts being at block boundaries, the timed processing is blocked by the run time of a block in the cyclic user program; the run time of the block is longer than the basic clock rate set in DX 0.	In the "ISTACK output" on the programmer, the error ID WEFEH is marked in the control bits.	The system program calls OB 33 as the user interface. If this is not loaded, the system program **continues program execution.**

Error reaction OB 33

In OB 33, you can program the required reaction to the interrupt collisions listed above. When OB 33 is called, the system program enters a collision ID in ACCU-1-L (bit nos. 0 to 9). You can see the meaning of these bits (bit = '1') in the following table.

Table 4-5 Timed interrupt collision IDs: meaning of the bits in ACCU-1-L

Bit number	Meaning
0	Queue overflow in timed interrupt period 1 (more than three timed interrupts are pending for OB 10).
1	Queue overflow in timed interrupt period 2 (more than three timed interrupts are pending for OB 11).
2	Queue overflow in timed interrupt period 3 (more than three timed interrupts are pending for OB 12).
3	Queue overflow in timed interrupt period 4 (OB 13 has been called again before the prior call was completely executed).
4	Queue overflow in timed interrupt period 5 (OB 14 has been called again before the prior call was completely executed).

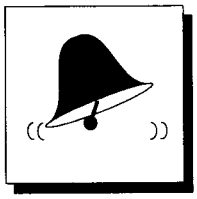
Bit number	Meaning
5	Queue overflow in timed interrupt period 6 (OB 15 has been called again before the prior call was completely executed).
6	Queue overflow in timed interrupt period 7 (OB 16 has been called again before the prior call was completely executed).
7	Queue overflow in timed interrupt period 8 (OB 17 has been called again before the prior call was completely executed).
8	Queue overflow in timed interrupt period 9 (OB 18 has been called again before the prior call was completely executed).
9	Timed interrupt clock pulse masked for too long.

After processing OB 33, the program is resumed at the interrupted timed interrupt OB.

Note
Following a collision of timed interrupts, the SAC does not point to the block at whose boundary the collision of timed interrupts took place (BE statement) but rather to the block which called the block that caused the error (the return address).

You can set the following parameters in data block DX 0 (refer to Chapter 7) for time-controlled program execution:
- setting the basic clock rate
- setting the clock distributor,
- setting priorities relative to interrupt-driven program execution,
- enabling/disabling timed interrupt processing.

4.5.4 Interrupt-Driven Program Execution



The following interrupt-driven program execution is possible with the CPU 948R:

- **PROCESS INTERRUPTS**
  via input byte IB 0 (max. 8 interrupts),

To service process interrupts, the default "process interrupts via IB 0 = on" must not be changed in the data block DX 0.

Program execution controlled by process interrupts means that a signal change in the input byte IB 0 causes the current program execution to be interrupted and a special program section to be executed.

> **Note**
> If you enable "servicing process interrupts via IB 0" you **cannot** use the delayed interrupt, the real time interrupt and the system interrupt.

Triggering

The signal state change of a bit in input byte IB 0 triggers the process interrupt.

User interfaces OB 2 to OB 9

If a process interrupt occurs, one of the OBs listed in the following table is called as the user interface.

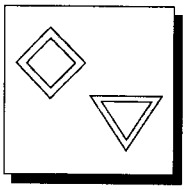
Table 4-6 User interfaces for process interrupts

Signal state change in IB 0 with bit	OB called
I 0.0	OB 2
I 0.1	OB 3
I 0.2	OB 4
I 0.3	OB 5
I 0.4	OB 6
I 0.5	OB 7
I 0.6	OB 8
I 0.7	OB 9

In the organization blocks OB 2 to OB 9, you program the part of your STEP 5 program to be executed when one of the process interrupts occurs indicated by a bit in input byte IB 0.
If the corresponding OB is **not loaded**, program execution is not interrupted. No interrupt-driven program processing takes place.

> **Note**
> If the I/O module no longer acknowledges when the CPU accesses IB 0, the system program recognizes a timeout and calls the user interface OB 28. If OB 28 is not loaded, the CPU changes to the stop mode.

Priority of process interrupts



The default setting means that PROCESS INTERRUPTS have a lower priority than the TIMED INTERRUPTS level.
A parameter in data block DX 0 allows you to change the default so that the PROCESS INTERRUPTS level has a higher priority than the TIMED INTERRUPTS level.

The following priorities are set for process interrupt processing:

I 0.0	OB 2	ascending priority
I 0.1	OB 3	
I 0.2	OB 4	
I 0.3	OB 5	
I 0.4	OB 6	
I 0.5	OB 7	
I 0.6	OB 8	
I 0.7	OB 9	

With process interrupts, **no** nested execution is possible. When a process interrupt OB has been completely processed and there are further process interrupts pending, the system program calls the OB with the next lower priority and processes it.
The PROCESS INTERRUPTS level is only exited when each signal change in input byte IB 0 has been dealt with and the corresponding OB completely processed.

> **Note**
> Process interrupt-driven program execution **cannot** be interrupted by further process interrupt-driven program execution.

Disabling interrupt-driven processing

An interrupt-driven program is called at a block or STEP 5 operation boundary in the cyclic program. This interrupt can cause problems when a cyclic section of program must be processed within a certain time (e.g. to achieve a certain reaction time) or when a series of operations must not be interrupted (e.g. when reading or writing interdependent values).

If a program section must **not** be interrupted by interrupt-driven processing, the following strategies are possible:

- Program this program section so that it does not contain a block change. Program sections that do not contain a block change can then not be interrupted.

- Use OB 122 with which you can disable the processing of process interrupts for a specific program section. Remember, however, that the timed interrupts are also disabled (refer to Section 6.3).

- Program the STEP 5 operation 'IA' (disable process interrupts). With the operation 'RA' (enable process interrupts) you can enable interrupt processing again.

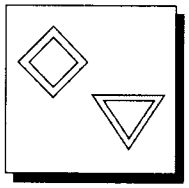
  Between these two operations, no process interrupt-driven program execution is permitted. The program section between these two operations **cannot** be interrupted.

- 'IA' and 'RA' are only possible in function blocks (extended operation set - refer to Section 3.5.4) and only apply to process interrupts via IB 0.

Note
If a process interrupt is disabled using OB 22 or delayed using OB 142, the RA operation is not effective.

Interrupt servicing can be disabled completely or separately for individual interrupts in data block DX 0. This is, however, only possible following a COLD RESTART (refer to Chapter 7), since DX 0 is only evaluated in a COLD RESTART.

Reaction time



The time required to react to a process interrupt/interrupt request corresponds to the processing time of a block. If, however, at the time the cyclic program is interrupted there are higher priority timed interrupts pending, the interrupt-driven program is only executed when all the pending timed interrupts have been completely processed.

The maximum reaction time between the occurrence and execution of a process interrupt/hardware interrupt is increased in this case by the processing time of the higher priority timed interrupts.

Program execution levels and flags

If you run your user program not only cyclically but also time and interrupt-driven, you run the risk of overwriting flags.
This is the case when the same flag areas are accessed at different program processing levels.

It is therefore advisable to assign flags to individual program processing levels and error OBs or at the beginning of time or interrupt-driven program execution, to "save" the signal states of multiply assigned flags in a data block and to write the values back at the end of the interrupt servicing. The same applies for a warm restart.

To avoid double use of flags, you can also use the S flags for most applications. Special "saving" strategies for flags are then no longer necessary, providing the S flags are assigned exclusively to individual program processing levels (there are enough S flags available).

Interrupt and Error Diagnostics

5

Contents of Chapter 5

5.1	Frequent Errors in the User Program	5 - 4
5.2	Error Information	5 - 5
5.3	Procedure for Error Analysis	5 - 9
5.4	Control Bits and Interrupt Stack	5 - 10
5.4.1	Control Bits	5 - 11
5.4.2	ISTACK Content	5 - 15
5.4.3	Example of Error Diagnosis using the ISTACK	5 - 20
5.5	Error Data Block (Error DB)	5 - 21
5.5.1	What is Entered in the Error DB?	5 - 21
5.5.2	Structure of the Error DB	5 - 22
5.5.3	Calling a User Interface	5 - 26
5.6	H Flag Word	5 - 27
5.7	Error Handling Using Organization Blocks	5 - 29
5.8	Causes of Error and Reactions of the CPU	5 - 32
5.8.1	OB 19: Calling a Logic Block That Is Not Loaded (KB)	5 - 33
5.8.2	OB 19: Opening a Data Block That Is Not Loaded (KDB)	5 - 33
5.8.3	OB 23/24, OB 28/29: Timeout Error (QVZ)	5 - 34
5.8.4	OB 25: Addressing Error (ADF)	5 - 35
5.8.5	OB 26: Cycle Time Exceeded Error (ZYK)	5 - 36
5.8.6	OB 27: (Substitution Error SUF)	5 - 37
5.8.7	OB 30: Parity Error and Timeout Error in the User Memory (PARE)	5 - 37
5.8.8	OB 32: Load and Transfer Error (TRAF)	5 - 38
5.8.9	OB 33: Collision of Timed Interrupts Error (WEFES/WEFEH)	5 - 39
5.8.10	OB 34: Error with G DB/GX DX (FEDBX)	5 - 41
5.8.11	OB 37: Entry in the Error DB	5 - 41

5.9 Self-Test ..... 5 - 43
5.9.1 Overview ..... 5 - 43
5.9.2 Organization of the Self-Test ..... 5 - 44

Interrupt and Error Diagnostics

5

This chapter explains how to avoid errors when planning and programming your STEP 5 programs.
You will see what help you can get from the system program for diagnosing and reacting to errors and which blocks you can use to program reactions to errors.
At the end of the chapter, you will learn how to activate integrated system functions for a self-test of the CPU 948R.

5.1 Frequent Errors in the User Program

The system program can detect effects of errors in the user program, faulty operation of the CPU, or errors in the system program processing.

The following list describes errors that occur most frequently during the start-up of the user program. You can avoid these errors by doing the following when you write you STEP 5 program.

- When specifying byte addresses for I/Os, make sure that the corresponding modules are plugged into the central controller or the expansion unit.
- Make sure that you have provided correct parameters for all operands.
- Be careful when changing function blocks. Check to see that the FBs/FXs are assigned the correct operands and that the actual operands are specified.
- Make sure that outputs, flags, timers, and counters are not processed in several locations in the program with operations that counteract each other.
- Make sure that timers are scanned only once per cycle (e.g., A T1).
- Make sure that all data blocks called in the program exist and are long enough.
- Check to see if all blocks called are actually in the memory.
- If required by other blocks (e.g. standard function blocks), scratchpad flags should be saved by interrupt-driven and time-controlled programs and loaded again when these program sections have been completed.

5.2 Error Information

If an error occurs during system start-up or during cyclic processing of your program, the sources of information described in this section can help you to find the problem. This includes:

- LEDs on the front panel of the CPU
- interrupt stack ISTACK and control bits
- block stack BSTACK
- Error DB
- H flag word

The following sections explain the ways of evaluating this information and how to use the information to analyze problems.

LEDs on the front panel of the CPU

If the CPU goes into the STOP mode when you do not want it to, check the LEDs on the front panel. They can indicate the cause of the problem.

LED display	Meaning
STOP LED lit continuously	The various states of the STOP LED indicate specific causes of interruptions and errors (see section 4.1).
STOP LED flashes slowly	
STOP LED flashes quickly	
SYS FAULT LED lit continuously	
ADF LED lit continuously	Addressing error
QVZ LED lit continuously	Timeout error
ZYK LED lit continuously	Cycle time exceeded error

PG online function OUTPUT ISTACK

You can get information about the status of the control bits and the contents of the interrupt stack (ISTACK) by using die PG online function OUTPUT ISTACK.

When the CPU goes into the STOP mode, the system program enters all the information it requires for a warm restart in the **ISTACK**. This information includes:

- contents of registers,
- contents of accumulators,
- STEP address counter SAC

  and

- results codes

Depending on where the interruption leading to the STOP occurred, the displayed information refers to user blocks or blocks of the system program (OB 0).

These entries are a valuable aid in error diagnosis.

Before the actual ISTACK is output on the programmer, the status of the **control bits** is displayed. The control bits mark the current operating status and specific characteristics of the CPU and the user program and provide additional information on the cause of an error.

You can call the ISTACK programmer function not only when the CPU is in the STOP mode, but also when it is in the RESTART or RUN mode. However, in the RESTART and RUN modes, you can only display the control bits (i.e., the first page of ISTACK information).

The meaning of the control bits and structure of the interrupt stack are described in detail in Section 5.4.

Online function OUTPUT BSTACK

You can display the contents of the block stack (BSTACK - see section 3.1.1 "Block nesting") using the PG online function OUTPUT BSTACK.

The BSTACK contains a listing of all blocks (blocks of the user program and organization block OB 0 of the system program) **called in sequence** and not completely processed when the CPU went into the STOP mode. Since the BSTACK is filled from the bottom, the top line of the BSTACK display contains the block containing the block that **called** the incorrect block.

In the **first** line, the information shown below is available:

Information	Meaning
BLOCK NO	Type and number of the block that called the faulty block
BLOCK ADDR	Absolute start address of the calling block in the program memory
RETURN ADDR	Absolute address of the first STEP 5 operation of this block in the user memory.
REL ADDR	Relative address (= difference "RETURN ADDR - BLOCK ADDR") of the next operation to be processed in the calling block. (You can display relative addresses on a programmer in the mode "disable input"/key switch and with S5-DOS from Stage IV upwards using the function key "addresses").
DB NO	Number of the last data block opened in the calling block
DB ADDR	Absolute start address of this block in the program memory (address of data word DW 0)

Example

Evaluating the BSTACK function:

BLOCK NO	BLOCK ADDR	RETURN ADDR	REL ADDR	DB NO	DB ADDR
PB 3	00090	98	00008		0
PB 2	00050	51	00001		0
PB 1	00040	41	00001		0
OB 1	00010	11	00001		0
OB 66 [1]	E2B10	E2C40	00130		0
OB 63	E0FC0	E12FA	0033A		0
OB 62	E0490	E0CBE	0082E		0
OB 61	E0010	E0273	00263		0

1) The blocks executed before OB 1 are internal blocks belonging to the system program (the BSTACK is structured chronologically).

```
In the example, PB 3 called the faulty block at relative address
"00008 - 1 = 00007".
During the jump to this faulty block, no data block was open.
```

Error DB

The system program enters all errors detected during the self-test and during program execution in the error DB along with an error class, the error number and the current date. The error DB also contains a static error image of all I/O addresses.

During system configuration with COM 155H, you specify a DB number for this data block (DX data block numbers are not allowed).

Using the COM 155H function "Diagnosis", you can output the error messages in the error DB in plain language, with the "Output DB" PG function you can read out the errors on the PG or evaluate them using your STEP 5 program.

H flag word

The H flag word provides you with important information about the status of your PLC in its "status "byte". Using its control byte, you can make certain settings with your STEP 5 program.

You specify the number of the H flag word with COM 155H.

5.3 Procedure for Error Analysis

If the CPU is in an abnormal stop mode, make use of all the information available to analyze the error, as follows:

Step	Action
1	Check the status of the STOP and SYS FAULT LEDs and the error LEDs on the front panel. These indicate causes of certain errors and interruptions.
2	Using the PG online function OUTPUT ISTACK, analyze the status of the control bits and the content of the ISTACK. This will provide you with further information about the location of the error and its cause.
3	Select the PG online function OUTPUT BSTACK: in the top line of the BSTACK display you will find information about the block which **called** the block causing the error.
4	Output the content of the error DB as plain language using the DIAGNOSIS function of COM 155 H.
5	Correct the errors in the ascending order so that secondary errors occurring later do not make error location more difficult.

Note
Correcting the error in step 5 of the table above is important with the S5-155H for the following reason:
Some errors would cause a CPU STOP in an S5-155U, but only cause an error message in the S5-155H because of the increased availability. Such errors are, however, serious and must nevertheless be corrected.

5.4 Control Bits and Interrupt Stack

You can use the online functions PLC INFO and OUTPUT ISTACK to analyze the following: operating status, characteristics of the CPU, characteristics of the user program, possible causes of errors and interruptions.

Note
You can display the **control bits** in **any** mode. You can display the **ISTACK** only in the **STOP** mode.

- The **control bits** indicate the current and previous operating status and the cause of the problem.
  If several errors occurred, the control bits indicate **all of them**.

- The **ISTACK** indicates the location of the interruption in question (addresses) with the current condition codes, the accumulator contents, and the cause of the problem.
  If several interruptions occurred, a multiple level ISTACK is constructed as follows (maximum 5 levels):

  DEPTH (level) 01 = last cause of interruption

  DEPTH (level) 02 = next to last cause of interruption, etc.

  When an ISTACK overflow occurs, the CPU goes into the STOP mode immediately (HARD STOP!). You must then turn the power off and on again and perform a cold restart.

The meanings of the individual abbreviations in the control bits and in the ISTACK are described below.

Note
The text on the screen of your programmer depends on the PG software you are using. It may therefore differ from the display shown here. The description of the screen information is nevertheless relevant.

5.4.1 Control Bits

When you display the ISTACK on your programmer, the status of the control bits is indicated on the first page (see Fig. 5-1).

Note
The ISTACK screen form shown in Fig. 5-1 reflects the PG software STEP 5/ST, Version 6.3 or STEP 5/MT Version 6.0 with the "Delta diskette CPU 948". In older versions of the PG software, the abbreviations of the control bits may be different. The meaning of the control bits, however, is as described in the following tables.

Control bits

System description:	EOVH	GEP	BATT	EINP	MEHRP	SYNCR
	X	X				
	TEST	BSTG	BEFG	MCG		
			X			
Stop cause:	PGSTP	HALT	STP	STS	STOPS	BEARBE
	UPROG	USYS	UANL	AFEL	SYSFHL	
Start-up IDs:	NEUDF	WIEDF	URLDF	NEUZU	WIEZU	URLER
	X					
	AWEG	ANEG	MSEG			
	X					
Error IDs:	QVZIN	PARIN	BSTKF	BSTEF	UMCG (BGRUN)	MODUN
	FE2S	SRAMF	UAFEHL	KDB1	KDX0	FDB1
	FDX0	FMODE	FEDBX	QVZNIO	WEFES	DB0UN

Fig. 5-1 Example of the first screen form page "OUTPUT ISTACK": control bits

You can output the control bits in every mode. They mark the current or previous status of the CPU and provide information on specific features of the CPU and your STEP 5 program.

The control bits listed under ERROR IDS mark errors that can occur in the RESTART (e.g., during an initial cold restart) and RUN (e.g., during time-controlled program processing) modes. If several errors occur, **all** errors are displayed in the control bits.

The following tables explain the meaning of the individual bits.

Table 5-1 Meaning of the control bits SYSTEM DESCRIPTION

SYSTEM DESCRIPTION	
Bit	**Meaning**
E0VH	Input byte IB 0 (process interrupts) exists, i.e. the digital input module addressed with '0' was plugged in during the last cold restart and the module acknowledged.
GEP	Programmable controller has a central back-up battery.
BATT	Battery failure in the central controller (BAU)
EINP	Single processor operation
MEHRP	Multiprocessing operation
SYNCR	Start-up of the CPUs in multiprocessing operation is synchronized
TEST	Test operation
BSTG	DX-0 setting "interrupts at block boundaries"
BEFG	DX-0 setting "interrupts at operation boundaries
MCG	Memory card inserted

Table 5-2 Meaning of the control bits STOP CAUSE

STOP CAUSE (see RS 7)	
Bit	**Meaning**
PGSTP	STOP mode set from programmer
HALT	Multiprocessor STOP mode: a) Selector switch on the coordinator (COR) is in the STOP position or b) Stop status caused by command STOP operation from system program when the corresponding error OB is not loaded and an error occurs
STS	STOP mode caused by STEP 5 operation 'STS' (after executing an operation)
STOPS	STOP mode caused by setting the mode selector to the STOP position
BEARBE	STOP mode after the PROGRAM TEST END programmer function

STOP CAUSE (see RS 7)	
Bit	**Meaning**
Table 5-2 continued:	
UPROG	STOP mode caused by user program
USYS	STOP mode caused by system program (warm restart possible)
UANL	STOP mode caused by illegal start-up type
AFEL	STOP mode caused by errors in the start-up block
SYSFHL	STOP mode caused by system error (may be caused by user error, e.g. overwriting system RAM with a block transfer or similar (when a system error is marked, a four-digit hexadecimal number/error code appears at the bottom edge of the screen - refer to RS 75 in Chapter 8).

Table 5-3 Meaning of the control bits START-UP IDS

START-UP IDs (see RS 8)	
Bit	**Meaning**
NEUDF	COLD RESTART was executed as last start-up type.
WIEDF	WARM RESTART was executed as last start-up type.
URLDF	Overall reset was executed or is active.
NEUZU	COLD RESTART permitted as next start-up type.
WIEZU	WARM RESTART permitted as next start-up type.
URLER	Overall reset required.
AWEG	AUTOMATIC WARM RESTART is preset.
ANEG	AUTOMATIC COLD RESTART is preset.
MSEG	Manual start is preset.

Table 5-4 Meaning of the control bits ERROR IDS

ERROR IDs	
Bit	**Meaning**
QVZIN	Timeout error in initialization
PARIN	Parity error in initialization
BSTKF	Wrong block ID
BSTEF	Wrong block delimiter
UMCG (BGRUN)	Illegal memory card inserted
MODUN	Content of the memory card too large for the available internal user memory
FE2S	Error on the second interface
SRAMF	System RAM error
UAFEHL	Error in the interrupt condition code word (UAW)
KDB1	No DB 1 in multiprocessing operation
KDX0	No DX 0 in multiprocessing operation
FDB1	Error in DB 1
FDX0	Error in DX 0
FMODE	No IB 0 process interrupts allowed in multiprocessor mode
FEDBX	Error in the STEP 5 operations G DB, GX DX
QVZNIO	QVZ test faulty
WEFES	Collision of software-driven timed interrupts: queue overflow
DB0UN	DB 0 has been changed since the last COLD RESTART. Therefore, no WARM RESTART possible.

5.4.2 ISTACK Content

If the CPU is in the stop state, you can display the content of the ISTACK on the screen after the control bit display by pressing the enter key. When the CPU goes into the STOP mode, the system program enters all the information it needs in this ISTACK for a warm restart.

You can use the entries in this ISTACK to see what kind of error occurred and where it occurred in the program.

If the stop state was caused by **a single** error, only **one** level of the ISTACK information is displayed. With **several** errors, the **corresponding number** of ISTACK levels are output (DEPTH 01, DEPTH 02, etc.). At all levels, only one error is marked as the CAUSE OF INTERRUPT.

If several errors have occurred DEPTH 01 marks the error detected immediately before the change to the stop state.

Fig 5-2 is an example of a PG display of the ISTACK content.

ISTACK

Depth 01

OP-REG:	1205	SAC (new):	000B3	DB-ADD:	00000	BA-ADD: 00108
BLK-STP:	EDEFF	PB-NO.:	9	DB-NO.:		OB-NO.: 1
PAGE		REL-SAC:	00013	DBL-REG:	0000	BR-REG: 00000
NUMBER:	00FD	SAC (old):	000B2	ICMK:	09DF3FBF	ICRW: FFFFFFFF
BRACKETS:	KE1 000	KE2 000	KE3 000	KE4 000	KE5 000	KE6 000
ACCU1:	0000 31BA	ACCU2:	0000 0005	ACCU3:	0004 0005	ACCU4: 0004 0005

CONDITION CODE:	CC1	CC0	OVFL	OVFLS	ODER	$\overline{\text{ERAB}}$
	STATUS	RLO				
	X	X				
CAUSE OF INTERR.:	KB	KDB	TRAF	SUF	STUEB	STUEU
	NAU	QVZ	ADF	PARE	ZYK	STOP
					X	
	STS	WEFEH	PEU	HALT		

Fig. 5-2 Example of a screen page "OUTPUT ISTACK"

Explanation of the ISTACK screen

DEPTH

Information level of the ISTACK when more than one error has occurred:

DEPTH 01 = last cause of stop to occur
DEPTH 02 = next to last cause of stop to occur
.....
DEPTH05 = ... (maximum depth)

Information about the error

The following table contains information about the ISTACK IDs with which the statement in the user program can be found which caused the CPU to change to the STOP mode.

Table 5-5 Meaning of the ISTACK IDs for errors

Information about the error	
ISTACK ID	**Meaning**
OP-REG	Operation register: Contains machine code (first word of the instruction processed last in an interrupted program processing level.
SAC (new)	STEP address counter (new): Contains the **absolute** address of the next operation in the **program memory** to be processed. When a warm restart occurs, the CPU continues the program with this operation.
DB-ADD	Absolute start address (DW 0) in the program memory of the data block currently opened (= 0000 if no data block was opened)
BA-ADD	Start addresses of the actual operands of a function block in the calling block.
BLK-STP	Block stack (BSTACK) pointer: Contains the 20-bit offset address of the last BSTACK entry (always Exxxx).
PB-NO (depending on type PB, OB ...)	Block type and number of the most recently **processed** block
DB-NO.	Number of the data block currently opened

Information about the error	
ISTACK ID	**Meaning**
Table 5-5 continued	
OB-NO. (depending on type OB, PB ...)	Block type and number of the last **calling** block
REL-SAC	Relative STEP address counter: contains the **relative** address (related to the block start address) of the next operation to be processed in the last block processed.
DBL-REG	Length of the data block currently opened
BS-REG	Content of the base address register prior to the transition to the stop mode
PAGE NUMBER	Number of the current dual-port RAM page selected (dual-port RAM access refers to this dual-port RAM page, you will find information about page access in Chapter 9)
SAC (old)	STEP address counter (old): contains the **absolute** address of the last operation processed in the **program memory** of an interrupted program level; if an error occurs, SAC (old) indicates the operation which caused the error.
ICMK	Interrupt condition code masking word: ICMK contains all the causes of interrupts which have occurred and not yet been completely processed.
ICRW	Interrupt condition code reset word
BRACKETS	Number of bracket levels: "KEx abc" where x = 1 to 7 levels, a = 'OR' (refer to bit codes), b =' RLO' (result of logic operation, refer to bit codes), c = 1: 'A(', c = 0: 'O('.
ACCU1 to ACCU4	Contents of the calculation registers (accumulators) at the time of the interruption

CONDITION CODE see Section 3.5

CAUSE OF INTERR. The following abbreviations (ISTACK IDs) indicate the most important causes of interruptions.

Table 5-6 ISTACK IDs CAUSE OF INTERRUPTION

CAUSE OF INTERR.	
ISTACK ID	**Meaning (called error OB)**
KB	Called block not loaded (OB 19)
KDB	Opened data block not loaded (OB 19)
TRAF	Load or transfer error (OB 32)
SUF	Substitution error (OB 27): Processed STEP 5 operation cannot be substituted
STUEB	Block stack overflow: Nesting depth too great; required action: COLD RESTART
STUEU	Interrupt stack overflow: Nesting depth too great; required action: POWER DOWN, POWER UP, then COLD RESTART
NAU	Power failure in the central controller
QVZ	Timeout (OB 23/OB 24/OB 28/OB 29)
ADF	Addressing error for digital inputs and outputs with process image (OB 25)
PARE	Parity error (OB 30)
ZYK	Cycle monitoring time exceeded (OB 26)
STOP	STOP mode caused by setting the mode selector to the STOP position
STS	STOP mode caused by STEP 5 operation 'STS' (after executing an operation)

CAUSE OF INTERR.	
ISTACK ID	**Meaning (called error OB)**
Table 5-6 continued:	
WEFEH	Collision of timed interrupts caused by the hardware clock (OB 33): timed interrupt clock was masked (ignored) for too long
PEU	I/Os not ready = power failure in expansion unit (the CPU does not go into STOP): After a statically pending PEU signal is removed, a message is entered in the error DB.
HALT	Not relevant for the CPU 948R

5.4.3
Example of Error Diagnosis using the ISTACK

Fig. 5-3 illustrates the structure of the ISTACK in conjunction with the interruptions that have occurred.

- The program execution level CYCLE (**OB 1**) is interrupted by an interrupt.
- The program processing level interrupt is then activated and **OB 3** called.
- A timed interrupt means that the INTERRUPT level is exited and the TIMED INTERRUPT level activated and **OB 13** processed.
- Incorrect addressing leads to the activation of the ADF level where **OB 25** is processed. In the error handling program, the user has programmed a stop operation (STS) the CPU aborts program execution.

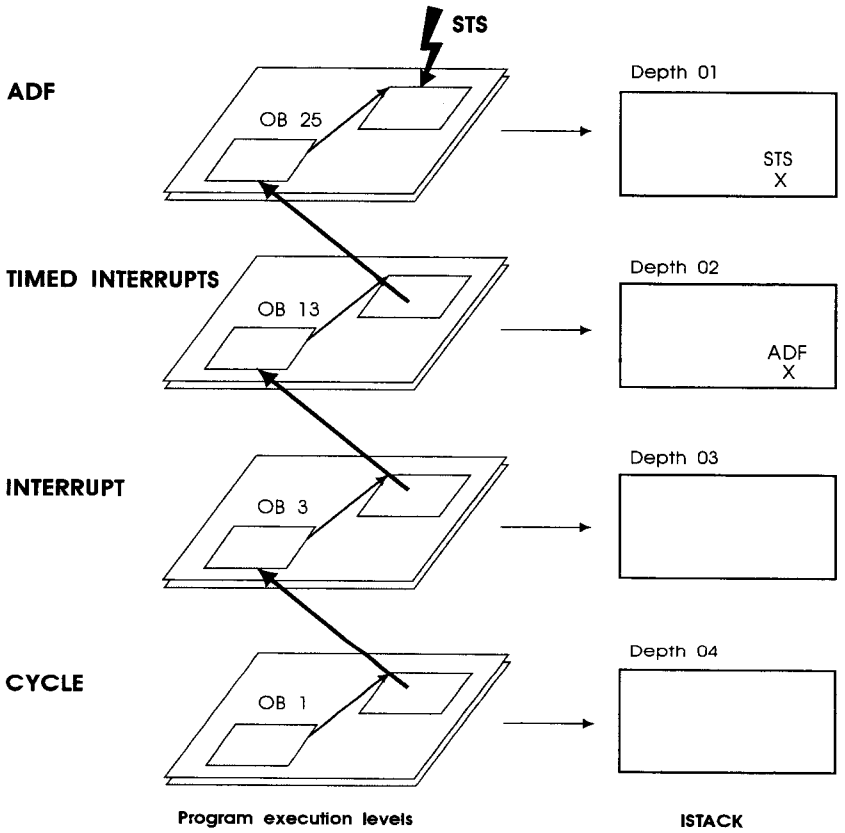


Fig. 5-3 Example of evaluating the ISTACK

Before the final transition to the stop mode, a total of four different program execution levels were interrupted. If you now display the ISTACK on the PG, you will obtain a **four-level** ISTACK, at the top the ISTACK with depth 01, with the ID of the **last** interrupted program execution level (= ADF). You can page down through the ISTACK until you reach depth 04, representing the CYCLE program execution level, which was interrupted **first.**

5.5 Error Data Block (Error DB)

During the system configuration with COM 155H, you specify the number of an error DB. You can use numbers from 3 to 255. During START-UP, the system creates a data block **DB** (not DX) with a standard length of 2K words for error entries.

5.5.1 What is Entered in the Error DB?

The following entries are particularly important if errors occur:

Error image

This is a static error image organized so that every repairable unit (CPU, parallel link, I/O modules, CP, IP) is assigned its individual bit. These bits are arranged according to continuous addresses or interface numbers. A further distinction is also made between units belonging to subunit A and/or subunit B or the switched I/Os.

All the bits of the error image are preset to '0'. If the system program recognizes an error during the self-test (see Section 5.9) or while the program is running, it sets the corresponding bit to '1'.

Status word

In addition to the entry in the error image, each error is assigned to a particular error group (e.g. "parallel link error", "I/O error"). These groups are entered in the **status word**: one group error bit is reserved for every group.

Each group error bit is set to '1' as long as there is one error entered for the particular group.

Error fields

The system program creates an **error field** consisting of 7 words for every error detected. Normally, every detected error is entered in the data block **once**.

Each error field contains among other things a **time stamp**. This consists of the current time in one of two selectable formats. When configuring, you can, however, also have the content of a selected "**H flag double word**" entered as the time stamp instead of the current time (see COM 155H).

Note
The content of the error DB is **deleted completely**

- when the master goes through a COLD RESTART,
- when you cancel the passive setting by setting the appropriate bit in the H flag word.

5.5.2 Structure of the Error DB

This section describes the structure of the error DB and the various entry fields.

Structure of the error DB

Fig. 5-4 shows the global structure of the error DB.

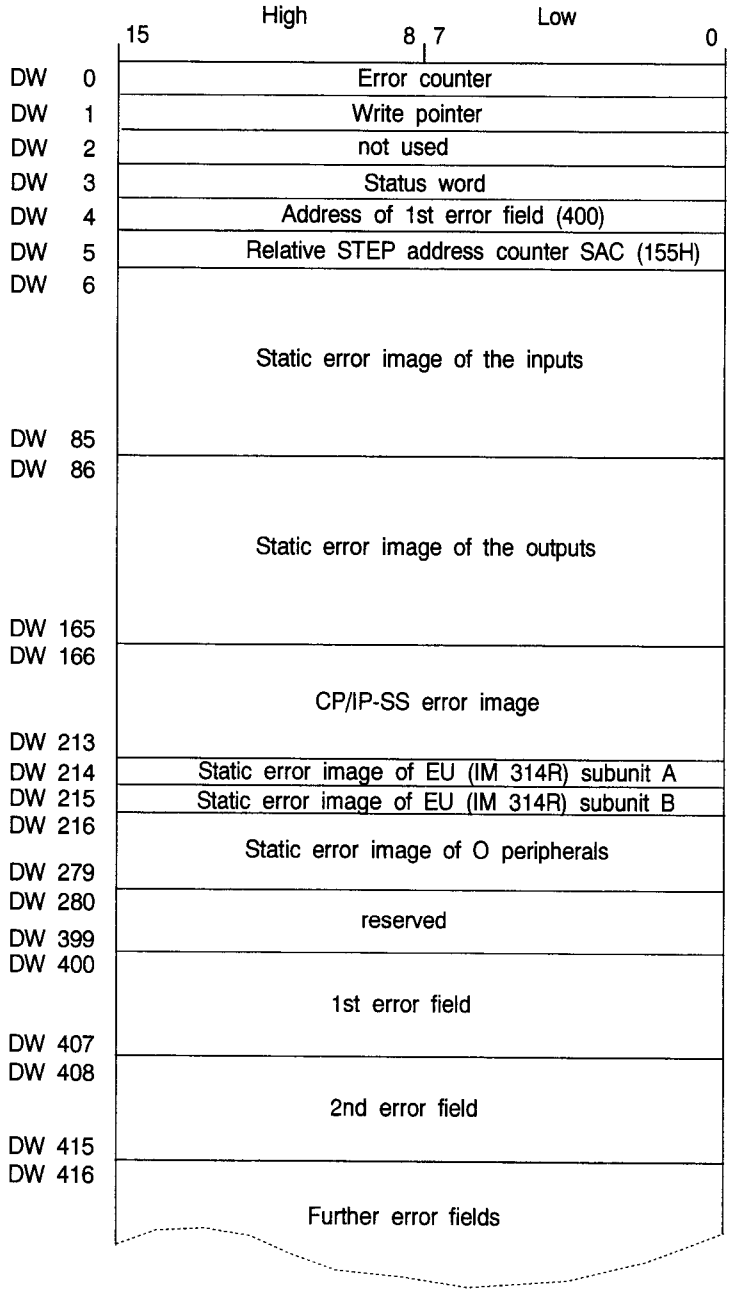


Fig. 5-4 Structure of the error DB

Meaning and structure of the entries in the error DB

The following section describes the various entries in the error DB.

Error counter (DW 0)

The errors are counted here. Once the 32767 errors is reached, the counter stops. The counter is reset by a COLD RESTART or when the passive setting is cancelled.

Write pointer (DW 1)

The write pointer always points to the start (the first data word number) of the current error field. The last error to occur is entered in the current error field. As long as there is no error in the error DB, the write pointer has the value '0'.

DW 2

Data word DW 2 is not used by the system program and is available for user entries.

Status word (DW 3)

H system errors can be divided into certain error groups. Each bit in the status word is assigned to such an error group. It is changed from '0' to '1' when the first error belonging to the group is entered in the error image and remains set to '1' as long as at least one error belonging to the group is present.

Table 5- 7 explains the meaning of the individual bits.

Table 5-7 Group error bits in the status word of the error DB

Bit no.	Abbrev.	Meaning
0	–	not used
1	CPU	Group error bit "CPU error"
2	–	not used
3	ANW	Group error bit "error in user memory" (memory card)
4	PK	Group error bit "parallel link error" (IM 304 – IM 324R)"
5	PEB	Group error bit "I/O bus error" (IM 304 – IM 314R)
6	PER	Group error bit "I/O error"
7	CP	Group error bit "CP/IP error" (page area)
8	PRJ	Group error bit "configuration error"
9	HAN	Group error bit "handling error"
10	SYS	Group error bit "system error"
11	HW	Group error bit "hardware fault" (e.g. defect CPU)
12	GER	Group error bit "device error" (e.g. POWER DOWN on EU)
13	MLD	message
14	–	not used
15	UMK	circulate ID

The circulate ID is set when all the error fields of the data block have been filled and the entries start again at the first error field. These earliest errors are then overwritten and can no longer be output.

Address of the first error field (DW 4)

This data word contains the start address of the first error field in the error DB: Data word number "400".

Relative SAC (DW 5 – only for maintenance specialists)

The data word contains the relative address (relative to the block start address) of the next operation to be executed in the last system program OB processed (e.g. I/O test OB). The last error entered was detected at this point.

Static error image of the inputs (DW 6 to DW 85)

Bit no. 0 in DW 6 corresponds to DI 0, bit no. 1 to DI 1 etc.

DW no.	Meaning
6 to 13	Static error image DI 0 to 127, subunit A
14 to 21	Static error image AI 128 to 255, subunit A
22 to 29	Static error image DI 0 to 127, subunit B
30 to 37	Static error image AI 128 to 255, subunit B
38 to 45	Static error image DI 0 to 127, switched
46 to 53	Static error image AI 128 to 255, switched
54 to 69	Error image of the O peripherals I 0 to 255, switched
70 to 85	Error image of the IPC flags I 0 to 255

Static error image of the outputs (DW 86 to DW 165)

Bit no. 0 in DW 86 corresponds to DQ 0, bit no. 1 to DQ 1 etc.

DW no.	Meaning
86 to 93	Static error image DQ 0 to 127, subunit A
94 to 101	Static error image AQ 128 to 255, subunit A
102 to 109	Static error image DQ 0 to 127, subunit B
110 to 117	Static error image AQ 128 to 255, subunit B
118 to 125	Static error image DQ 0 to 127, switched
126 to 133	Static error image AQ 128 to 255, switched
134 to 149	Error image of the O peripherals Q 0 to 255, switched
150 to 165	Error image of the IPC flags Q 0 to 255

CP/IP-SS error image (DW 166 to DW 213)

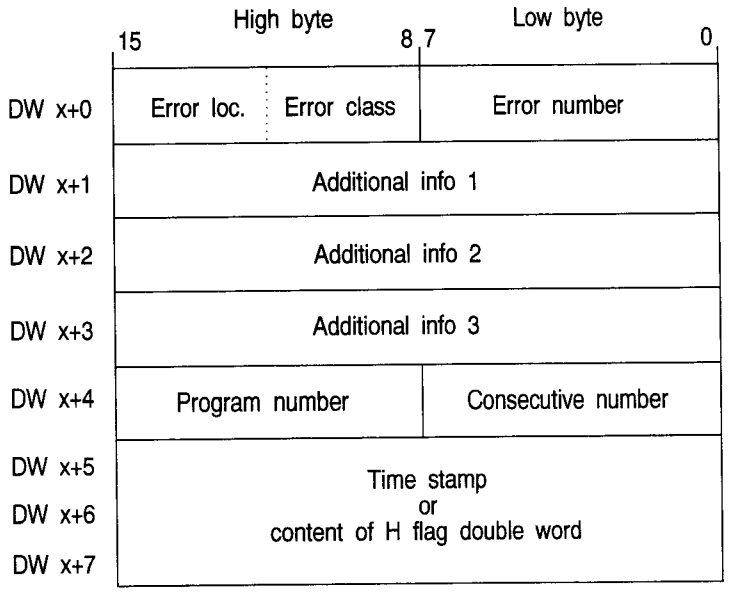
DW no.	Meaning
166 to 181	CP/IP error image 0 to 255, subunit A
182 to 197	CP/IP error image 0 to 255, subunit B
198 to 213	CP/IP error image 0 to 255, switched

Error image of the O peripherals (DW 216 to DW 279)

DW no.	Meaning
216 to 231	Static error image DI/AI 0 to 255 Q, subunit A
232 to 247	Static error image DI/AI 0 to 255 Q, subunit B
248 to 263	Static error image DQ/AQ 0 to 255 Q, subunit A
264 to 279	Static error image DQ/AQ 0 to 255 Q, subunit B

Error field

Fig. 5-5 shows the structure of an error field.



x = start address of the error block

Fig. 5-5 Structure of an error field

Error class and error location

Bit no.	15			12	11			8
	B	A	RES	MA	–	PAS	WST	HST

- Location:

  MA Error occurred in the master

  RES Error occurred in the standby

  A Error occurred in subunit A

  B Error occurred in subunit B

- Error class:

  The reaction to the error is entered here (default standard reaction).

  HST HARD STOP (e.g. with CPU error)

  WST SOFT STOP (e.g. with signature error in user program etc.)

  PAS Passivation (e.g. with QVZ, discrepancy error etc.)

  If, for example, a test error is entered in the error field, bit no. 1 is set too '1', since the standard reaction is a change to the SOFT STOP.

Error number:

The error number is a consecutive number from 1 to 255; each error number has a plain language text assigned to it (see "Structure of the user message field" in this section).

The meaning of the error numbers can be found in a table in the appendix.

Additional information / user message field / time stamp:

For detailed information, refer to Part II of Volume 1 of this manual.

5.5.3 Calling a User Interface

After every entry in the error DB, the system program calls **OB 37** as the user interface, if it is loaded. Here, you can write a STEP 5 program to evaluate the error DB.

5.6 H Flag Word

The H flag word contains important information about the status of your PLC (e.g. "PLC in error search mode"). You can select the number of the H flag word from 0 to 254 and specify it during configuration with COM 155H.

Structure

The H flag word consists of a **status byte** (high byte) and a **control byte** (low byte). The information in the status byte can be read bit-by-bit by the user program, the control information in the control byte can be set bit-by-bit.

Status byte

The bits in the status byte have the following meaning:

Table 5-8 Status byte in the H flag word

Bit no.	Abbrev.	Meaning
0	SOL	Bit = '1': PLC is in the mode "solo mode" Bit = '0': other mode
1	RED	Bit = '1': PLC is in the mode "redundant mode" Bit = '0': other mode
2	MAS	Bit = '1': PLC is in the mode "master" Bit = '0': other mode
3	ANK	Bit = '1': PLC is in the mode "coupling standby" Bit = '0': other mode
4	ZGA	Bit = '1': central controller is subunit A Bit = '0': central controller is subunit B (in the redundant mode this bit is always '0')
5	AMA	Bit = '1': subunit A is master Bit = '0': subunit B is master
6	ADL	Bit = '1': "standby being updated" Bit = '0': other mode
7	AGF	Bit = '1': PLC error occurred in switched I/Os Bit = '0': no error

Control byte

The bits of the control byte have the following meaning:

Table 5-9 Control byte in the H flag word

Bit no.	Abbrev.	Meaning
0	OAT	Bit = '1': restart master without test Bit = '0': restart master with test OAT is reset by the system program at a COLD RESTART and after OVERALL RESET
1	DPO	Bit = '1': cancel passive setting **without** deleting the error fields Bit = '0': no action (default) The bit is deleted by the system program after cancelling the passivation.
2	ARE	Bit = '1': updating standby disabled Bit = '0': updating standby enabled (default)
3	–	reserved
4	DPA	Bit = '1': cancel passivation Bit = '0': do not cancel passivation default)
5	AUM	Bit = '1': request switchover Bit = '0': no switchover request (default) AUM is reset by the system program after a switchover.
6	RST	Bit = '1': set standby to STOP Bit = '0': do not set standby to STOP (default) RST is reset by the system program after it is executed
7	–	reserved

For more information about the meaning of the H flag word, refer to Part II of Volume 1 of this manual.

5.7 Error Handling Using Organization Blocks

When the system program detects an error, it calls the appropriate organization block to handle it. You can determine further operation of the CPU by programming the appropriate organization block. Therefore, the CPU can do one of the following:

- continue normal program processing
- go into the STOP mode

  and/or

- process a special "error handling program"

For the following causes of error, OBs are available:

Table 5-10 The organization blocks called in case of errors

Cause of error	Organization block called	Reaction of CPU if OB is not programmed
Call of a block that is not loaded (KB)	OB 19	none
Attempt to open a data block DB/DX that is not loaded (KDB)	OB 19	STOP
Timeout in the user program during access to I/O peripherals (QVZ)	OB 23	none
Timeout during update of the process image	OB 24	none
Addressing error (ADF)	OB 25	STOP [1)]
Cycle time exceeded (ZYK)	OB 26	STOP
Substitution error (SUF)	OB 27	STOP
Timeout by reading input byte IB 0 (process interrupts – QVZ)	OB 28	STOP
Timeout during access to the distributed I/O peripherals (extended address area — QVZ)	OB 29	none
Parity error and timeout in the user memory (PARE)	OB 30	STOP

Cause of error	Organization block called	Reaction of CPU if OB is not programmed
Table 5-9 continued:		
Load and transfer error (TRAF)	OB 32	STOP
Collision of timed interrupts: a) queue overflow (control bit WEFES) b) timed interrupt clock was masked (ignored) for too long (WEFEH)	OB 33	 STOP none
Error during STEP 5 operation "G DB/GX DX" (control bit FEDBX)	OB 34	STOP
Entry in error DB (refer to Section 5.5)	OB 37	none

[1] The CPU changes to the STOP mode only if the addressing error is not disabled by the STEP 5 operation "IAE".

*Examples of reactions to organization blocks which are **not loaded***

a) **No** reaction; cyclic program processing is **not** interrupted.

If a timeout error occurs and neither OB 23 nor OB 25 is loaded, cyclic program processing is **not** interrupted according to the table above. The CPU does not react.
If you want the CPU to go into the STOP mode when a timeout error occurs, you must enter a stop statement (STP for STOP at cycle end) in the appropriate organization block (e.g. OB 23 with QVZ) and terminate it with the block end statement 'BE'.

Example of OB 23:

```
:          QVZ has occurred
:
:STP       Cyclic processing is aborted
:BE        CPU changes to the stop mode
```

b) Reaction : the CPU changes to the **STOP mode**.

The CPU changes to the STOP mode immediately when a corresponding error (e.g. cycle or load/transfer error) occurs - if you did not load the appropriate organization blocks.

If, as an exception, you do not want one of these errors to interrupt cyclic program processing (e.g. while putting the system into operation), a block end statement in the appropriate organization block is sufficient.

Example of OB 25:

```
:          ADF occurred
:
:BE        Cyclic processing is continued, no CPU STOP
```

Interruptions during processing of error organization blocks

After the system program calls the appropriate organization block, the user program in that block is processed.
If another error occurs while that organization block is being processed, the program is interrupted at the next operation boundary and the appropriate organization block is called, just as in cyclic program processing.

The system program processes organization blocks in the order in which they are called.

Note
You can nest a maximum of five error organization blocks. With more than 5 errors, the CPU goes into the HARD STOP mode because of ISTACK overflow.

5.8 Causes of Error and Reactions of the CPU

Specific events can interrupt cyclic, time-controlled, or interrupt-driven program processing at operation boundaries when the CPU is in the RUN mode.

During initialization and also in the RESTART mode, interruptions can stop the start-up program and put the CPU into the STOP mode. The CPU then changes to the stop mode and calls the organization block for this particular error. Interruptions during the start-up program are handled like those in the RUN mode.

The reaction depends on the cause of the interruption:

- immediate change to the STOP mode, without calling the error OB (e.g. NAU → hard stop, STUEU → hard stop, PEU → soft stop),
- before changing to the STOP mode, the system program calls an error OB which you can program and (depending on the cause of the error) avoid a change to the stop mode (e.g. QVZ/IB 0 → OB 28, ADF → OB 25).

If an error occurs, note the entries in the control bits under "Error IDs" and the entries in the ISTACK under STOP CAUSE

Note
Points to note with the S5-155H with T PY/T PW and addressing errors:

If there is a QVZ and ADF (when the addressed I/Os are not configured or do not exist) when one of the operations above is being executed, only QVZ is signalled in the solo mode and both QVZ and ADF in the redundant mode.

The following sections explain possible causes of error in greater detail.

5.8.1 OB 19: Calling a Logic Block That Is Not Loaded (KB)

If your program jumps to a block that does not exist, the system program detects an error. This applies to all logic blocks and also for conditional and unconditional calls.

When the system program detects the call of a logic block that is not loaded, it calls **OB 19**, if this is loaded. In OB 19, you can specify how the CPU should proceed.

If OB 19 does not exist, the system program continues executing the interrupted STEP 5 program at the next operation.

5.8.2 OB 19: Opening a Data Block That Is Not Loaded (KDB)

If you open a data block or an extended data block in your program that does not exist in the memory or is marked as invalid, the CPU detects an error and the system program calls **OB 19**, if this is loaded. If OB 19 is not loaded, the CPU changes to the STOP mode. A zero is entered in the DBA and DBL registers.

Note
OB 19 is called both when a logic or data block is not loaded. You can read system data register RS 75 to determine (via the STEP 5 program) which type of error occurred. The contents of RS 75 are as follows:
- for a KB error: 0101H,
- for a KDB error: 0904H.

5.8.3 OB 23/24, OB 28/29: Timeout Error (QVZ)

A timeout error occurs when an addressable memory area does not respond to write or read accesses with the ready signal ("RDY") within a specific time after being addressed. This time is monitored by the hardware. A defective module or the removal of a module during operation of the programmable controller can cause a timeout error.

The following timeout errors interrupt the user program, jump to system program error handling, and call the appropriate blocks if they are loaded:

OB 23

QVZ with direct I/O access:

Cause of error	Reaction to error
Timeout error in the user program during direct access via the S5 bus to an IP (e.g., with load and transfer operations "L/T P..." or "L/T Q...").	If OB 23 is not loaded, the system program continues the processing of the user program.

OB 24

Cause of error	Reaction to error
Timeout error during update of the process image input/output tables.	If OB 24 is not loaded, the system program continues processing of the user program.

Extension of the execution time

With calling OB 23 or OB 24 a timeout error increases the execution time of the STEP 5 operation which caused the timeout when the program is resumed:
extension = "acknowledgement monitoring time + time of error handling in the system program + processing time if error OB is called".

OB 28

Cause of error	Reaction to error
Timeout error at input byte IB 0 (process interrupts)	If OB 28 is not loaded, the CPU changes to the STOP mode.

OB 29

Cause of error	Reaction to error
Timeout error of the distributed peripherals in the following address areas: - F 0000H to F EFFFH - F F200H to F FFFFH	If OB 29 is not loaded, the system program continues processing of the user program.

Error address

When a timeout error occurs, you can read the error address in the system data area (see Chapter 8):

RS	Contents	Address
68	QVZ error address high	E F044H
69	QVZ error address low	E F045H

5.8.4 OB 25: Addressing Error (ADF)

An addressing error occurs when a STEP 5 operation references a process image input or output to which no I/O module was assigned at the time of the last COLD RESTART (the module is not plugged in, it is defective, or it is not defined in data block DB 1 of the CPU.

The STEP 5 operation at which the addressing error occurred is processed completely: For bit operations, the bit in the process image is scanned and combined logically or set/reset. Load and transfer operations are also executed. Continued processing can, however, result in incorrect or unwanted reactions.

When an addressing error occurs, the system program interrupts further processing of the user program and calls organization block **OB 25**. After running the program contained in OB 25, the program is resumed at the next operation.
If OB 25 is not loaded, the CPU changes to the STOP mode with an addressing error.

The STEP 5 IAE operation disables addressing error monitoring for individual program parts or for the entire program. You can enable it again using the RAE operation (see Section 3.5.4 and List of Operations).

5.8.5 OB 26: Cycle Time Exceeded Error (ZYK)

The cycle time is the time between the start of one OB 1 and the next. It includes the entire duration of cyclic program processing including interrupts, interrupt servicing and system program activities. The cycle monitoring time set on the CPU can, for example, be exceeded by incorrect programming (program loop).

The cycle time required is extended by the time taken for a PG function.

> **Note**
> Hardware faults as the cause of cycle time errors are **extremely** rare. Normally, the error is in the user program or the programs and cycle monitoring time are incompatible.

When a cycle time exceeded error (ZYK) occurs, the **standby** always changes to the stop mode. The cycle monitoring time of the master CPU is started. The H system continues operation in the solo mode.

If a further cycle time exceeded error occurs, the system program interrupts the user program and calls **OB 26** if this is loaded. The monitoring time is then restarted (triggered). If the monitoring time is exceeded again, before OB 26 is completed, the master CPU changes to the stop mode.
If OB 26 is not loaded, the **master CPU** also changes to the STOP mode.
When you program OB 26, it should always include a reaction to errors and **not be left empty** (only BE operation) so that the error detection and localization capabilities of your S5-155H are not reduced.

The cycle monitoring time is variable (10 to 2550 msec) and is retriggerable (see above).
You can specify the cycle monitoring time individually by making an entry in DX 0 (refer to Chapter 7) or by programming OB 31. The default monitoring time is 200 ms.

In the cyclic program, the cycle monitoring time can be retriggered by calling the special function OB 222.

5.8.6 OB 27: (Substitution Error SUF)

If an operation with a formal operand is to be carried out in a function block, the CPU replaces (substitutes) this formal operand with the actual operand in the block when the block is called during user program processing.

If the CPU detects an illegal substitution, it interrupts the user program and calls **OB 27**, if this is loaded. If OB 27 is not loaded, the CPU changes to the STOP mode.

Apart from an illegal substitution, SUF is also indicated in the following situations:

- illegal operation code,
- special situation:
  you cannot open data blocks DB 0 and DB 1. The CPU handles the operations "C DB 0" and "C DB 1" like substitution errors. A zero is entered in the DBA and DBL registers.

5.8.7 OB 30: Parity Error and Timeout Error in the User Memory (PARE)

The user memory is protected by a parity bit. The system program checks whether the parity bit is correct each time the user memory is accessed. If the parity bit is incorrectly set, a parity error is indicated.

The system program calls **OB 30**. If OB 30 is not loaded, the CPU changes to the STOP mode.

The same reaction takes place if a timeout error occurs in the user memory.

PARE accessing the operating system RAM

If a parity error occurs when accessing the operating system RAM, the system program does **not** call OB 30, but **changes to a HARD STOP**.

Error address

If a parity error occurs, the address which caused the error can be read out of the system data area (refer to Chapter 8):

RS	Contents	Address
70	error address high	E F046H
71	error address low	E F047H

5.8.8 OB 32: Load and Transfer Error (TRAF)

A load and transfer error is indicated in the following situations:

- When accessing data in data blocks or extended data blocks, the CPU compares the length of the opened block to the parameter in the load or transfer operation.
  If the specified parameter exceeds the actual data block length, the CPU does not execute the load or transfer operation. This prevents data in the memory from being overwritten by mistake during transfer operations. With load errors, the contents of the accumulator are retained.

- A load or transfer error is also detected if a single bit within a non-existent data word is to be scanned or changed.

- If no data block has yet been opened (with "C DBn" or "CX DXn") prior to access to a data word, this also causes a load/transfer error.

- Accessing the memory using incorrect absolute addresses via the BR register or incorrect area boundaries with the "TNW", "TXW" and "TXB" operations can cause a load or transfer error.

When the system program detects a load or transfer error, it calls **OB 32**, if this is loaded. The operation that caused the load or transfer error is not processed.
If OB 32 is not loaded, the CPU changes to the STOP mode.

If a load/transfer error particular to the **H system** occurs (= field transfer to redundant I/Os or byte read access to redundant analog inputs), and OB 32 is not loaded, then **only the standby CPU changes to the STOP mode**, the master continues in the solo mode.

5.8.9 OB 33: Collision of Timed Interrupts Error (WEFES/WEFEH)

Time-controlled program processing (timed interrupts) is handled by organization blocks OB 6, OB 9 and OB 10 to OB 18.

The following types of timed errors can occur on the CPU 948R:

Queue overflow

Cause:

Queue overflow servicing timed interrupts:

- there are more than three timed interrupts pending for one of the three shortest periods (OBs 10 to 12)

  or

- one of the other OBs (OBs 13 to 18) has been called before the first call was completely processed.

Reaction:

The system program calls **OB 33** as the user interface, if this is loaded. You can program the reaction to this state in this OB.

If OB 33 is not loaded, the CPU changes to the STOP mode.

PG display in "OUTPUT ISTACK":

The bit WEFES is marked in the control bits.

Masking the timed interrupt clock

Cause:

The internal timed interrupt clock is masked (ignored) too long (applies to interruptions at block boundaries/process interrupts).

This situation is related to the basic clock rate of the internal timed interrupts and the scan time of a block in the cyclic user program. If the scan of a cyclic block runs longer than the basic clock rate, a collision of timed interrupts occurs.

Reaction:

The system program calls **OB 33** as user interface, if this is loaded. Here, you can program the reaction to this state.

If OB 33 is not loaded, the CPU continues processing the program.

PG display with "OUTPUT ISTACK":

The bit WEFES is marked in the control bits.

When the system program calls OB 33, a code for the collision of timed interrupts is transferred to ACCU-1-L (see Section 4.5.3).

Note
In the "process interrupt via IB 0" mode **and** "interruptability at block boundaries" the step address counter (SAC) does not point to the block **at whose boundary (BE statement) the collision of timed interrupts** took place. It points to the block that **called the block that caused the error** (return address).

As long as an error is pending or reoccurs every time the STEP 5 operation in question is processed in each scan, the appropriate error organization block is always called.

This can increase the cycle time considerably, depending on the duration of error handling by the system program and of processing time of the organization block.

STP in OB 33

If there is a collision of timed interrupts in the redundant mode and if there is a 'STP' operation programmed in the loaded OB 33, then **only the standby CPU changes to the STOP mode**, the master continues in the solo mode.
If a further collision occurs and if a 'STP' operation is programmed in OB 33, the master CPU also changes to the STOP mode.

5.8.10 OB 34: Error with G DB/GX DX (FEDBX)

Causes:

- With the operation G DB/GX DX, an illegal block number was specified (number of a reserved block, number > 255),
- With the operation G DB/GX DX, an illegal block length was specified (> 4091),
- There is no longer enough space in the user memory for the operation GDB/GXDX.
- The operation G DB/GX DX is not permitted at this time since the standby linking is currently active.

Reaction:

The system program calls **OB 34** is this is loaded. If OB 34 is not loaded, the CPU changes to the stop mode.

5.8.11 OB 37: Entry in the Error DB

Since the system program automatically calls OB 37 after every entry in the error DB, it makes sense to evaluate the content of the error DB with a STEP 5 program (using the error counter, write and read pointers and status word etc.), so that the required reaction to an error can be triggered,
An example of such a STEP 5 program can be found on the following page.

Example

```
The following program shows an example of a reaction to a specific error:

FB 37 examines the last entered error field for the error number 'yy'. If
this error number is found, the reaction to this specific error can be
triggered.

OB 37                                                          LEN=yy

SEGMENT 1
0000       :JU   FB 37
0001
0002 NAME  :ERR-EVAL
0003       :BE

FB 37                                                          LEN=yy

SEGMENT 1          0000
NAME:ERR-EVAL

0005       :MBA                          Error address from BR register
0006       :LRW +1                       Pointer to current error field
0008       :+D
0009       :MAB
000A       :LRW +0                       Load error
000C       :L    KB 255                    number
000D       :AW
000E       :L    KB yy                   Error
000F       :!=F                            number 'YY' ?
0010       :JC   =M001
0011       :JU   =BE
0012 M001  :                             Here, you can program the
0013       :                             reaction to error
0014       :                             number 'yy'.
0015       :
00xx BE    :BE
```

5.9 Self-Test

5.9.1 Overview

The CPU 948R contains integrated self-test routines in the system program. These routines detect and localize hardware faults and indicate which modules are faulty and must be replaced.

What is tested?

The self-test routines check the following system components:

Which component?	What for (when)?
CPU	STEP 5 operations, timers, interrupt mask, cycle time monitoring etc.
EPROM, RAM	e.g. comparison of RAMs on the CPUs of both subunits; checksum test of OBs, SBs, PBs and FB and FX data blocks
I/O bus with IM 314R	Short circuits and breaks on the I/O bus cable 721 to the IM 314R are detected
Page addressing	In the cyclic mode, the page addressing is tested once per complete test run. The test detects the following incorrect functions: - A CP reacts to (acknowledges) not only its own interface number but also, the remaining 255 interface numbers. - An unused interface number acknowledges. All unused interfaces are checked. This test is also performed once per START-UP.
Parallel link IM 304/IM 324R	The dual-port RAM of the parallel link is checked by both subunit A and subunit B. This test reveals short circuits and breaks.
Locating facilities (LF)	The LF are tested once every 10 hours The LF of the redundant DIs and DQs for constant 0 errors and (only in START-UP) for constant 1 errors.

5.9.2 Organization of the Self-Test

The whole self-test of the S5-155H consists of approximately 3000 test slices.

Configuring the test slices for the cycle

You can configure the number of test slices processed once per cycle (see COM 155H, operating system parameter assignment). The number of test slices per cycle can be set to a value between 1 and 19. This corresponds to a test execution time of 2 to 38 ms. A longer test execution time reduces the error detection time.

Handling pending interrupts

If more than one test slice is configured the system program checks every 2 ms to find out whether or not a timed or process interrupt is waiting for servicing. If there is an interrupt, the corresponding OB is processed before the next test slice. The self-test execution time is then extended by the time required to process the interrupt.

Self-test in the various operating modes

Different self-test routines are called in the in the CPU modes listed below:

Self-test in START-UP

During the START-UP of a CPU, the complete self-test is executed. If the self-test routine detects an error here, the CPU changes to the STOP mode. An error message is entered in the error DB.

During the self-test in START-UP, the RUN and STOP LEDs on the front panel are lit constantly.

The self-test during START-UP is not dependent on the type of start-up being run through.

Self-test in the RUN mode

After OB 1 has been processed, i.e. once per cycle, **part of the test routines are run through at short intervals** (2 ms test slices). The test routine runs transparently in the background for the remaining software until a hardware failure is detected.

Self-test in error search mode

If an error cannot be assigned to a particular subunit, the standby PLC changes to the error search mode. in this mode, the **whole self-test** is run through taking between 10 and 30 seconds.

Integrated Special Functions

6

Contents of Chapter 6

6.1	Introduction	6 - 4
6.1.1	Interfaces	6 - 5
6.1.2	Error Handling	6 - 6
6.1.3	Notes on the Special Functions OB 124, OB 125 and OB 254/255	6 - 7
6.2	OB 121: Set/Read System Time	6 - 8
6.3	OB 122: "Disable Interrupts" On/Off	6 - 12
6.4	OB 124: Delete STEP 5 Blocks	6 - 14
6.5	OB 125: Generate STEP 5 Blocks	6 - 17
6.6	OB 131: Delete ACCUs 1, 2, 3 and 4	6 - 20
6.7	OB 132/133: Roll-Up ACCU/Roll-Down ACCU	6 - 21
6.8	OB 141: "Disable Single Cyclic Timed Interrupts" On/Off	6 - 23
6.9	OB 142: "Delay All Interrupts" On/Off	6 - 26
6.10	OB 143: "Delay Single Cyclic Timed Interrupts" On/Off	6 - 29
6.11	OB 150: Set/Read System Time	6 - 32
6.12	OB 151: Set/Read Time for Real Time Interrupt	6 - 37
6.13	OB 153: Set/Read Time for Delayed Interrupt	6 - 44
6.14	OB 222: Restart Cycle Monitoring Time	6 - 47
6.15	OB 254/255: Copy/Duplicate Data Blocks	6 - 48

Integrated Special Functions

6

The following chapter describes the special functions integrated in the system program, where you can use these functions and how to call and assign parameters to the special function OBs.
You will also learn how to recognize errors in the execution of a special function and possible ways of handling them in the program.

6.1 Introduction

The operating system of the CPU 948R provides you with special functions which you can call if necessary with a conditional (JC OB x) or an unconditional (JU OB x) block call. Organization blocks OB 120 to 255 are reserved for these special functions.

These functions are known as **integrated** special functions, since they are a fixed part of the system program. As user, you can call these special functions but cannot read or modify the corresponding program.

The following table provides an overview of the existing special functions.

Table 6-1 Overview of the special functions available with the CPU 948R

Block	Function	Refer to section/page
OB 121	Set/read system time (compatible with CPU 946/947)	6.2/6 - 8
OB 122	"Disable interrupts" on/off	6.3/6 - 12
OB 124 OB 125	Delete STEP 5 blocks Generate STEP 5 blocks	6.4/6 - 14 6.5/6 - 17
OB 131	Delete ACCU 1, 2, 3 and 4	6.7/6 - 25
OB 132 OB 133	Roll-up ACCU Roll-down ACCU	6.8/6- 26 6.8/6- 26
OB 141 OB 142 OB 143	"Disable single cyclic timed interrupts" on/off "Delay all interrupts" on/off "Delay single cyclic timed interrupts" on/off	6.9/6 - 28 6.10/6 - 31 6.11/6 - 34
OB 150 OB 151 OB 153	Set/read system time Set/read time for real time interrupt Set/read time for delayed interrupt	6.12/6 - 37 6.13/6 - 42 6.14/6 - 49
OB 222	Restart cycle monitoring time	6.16/6 - 52
OB 254, 255	Copy/duplicate DB and DX data blocks	6.18/6 - 53

6.1.1 Interfaces

The following are available as interfaces to the special functions:

Block call

- Conditional/unconditional block call JC .. / JU ..

Parameters

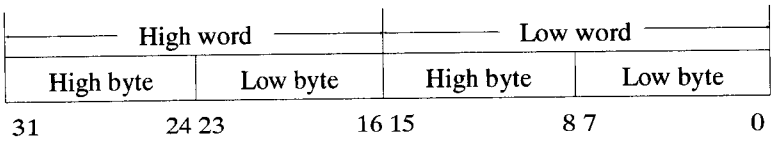
- Parameters for defaults via ACCU 1 and possibly ACCU 2 and/or memory locations

  In the following description of the individual special functions, all the data required by the CPU to execute the special function correctly are listed under the term **parameters**. Before calling the special function in the STEP 5 program, you must load this data in the accumulators or in the specified memory locations.

Writing to ACCUs

When assigning parameters for the special function organization blocks, please note the following conventions for writing to the ACCUs:

ACCU 1:	ACCU 1,	**32 bits**
ACCU-1-L:	ACCU 1, low word,	**16 bits**
ACCU-1-LL:	ACCU 1, low word, low byte,	**8 bits**
ACCU-1-LH:	ACCU 1, low word, high-byte,	**8 bits**



6.1.2 Error Handling

An error occurring in the execution of the active special function triggers a special error reaction in the system program.

In terms of this error reaction by the system program, two groups of special functions can be distinguished.

- **Group 1:**

ACCU condition code bits

Group 1 includes all the special functions with which IDs are transferred to ACCU 1 if an error occurs to further explain the error.

- **Group 2:**

RLO, CC0/CC1

With some of the special functions, the RLO or the status bits CC0/CC1 are written to indicate errors for specific special functions.
If an error occurs when using one of these special functions, the RLO is set to "1" in most cases. In your STEP 5 program, you can use a JC operation (conditional jump) to evaluate the RLO for these special functions and then react to an error.

In some special functions, the results bits CC0 and CC1 are affected by the processing of a special function. You can scan these bits in your STEP 5 program using a compare operation and once again program a reaction to an error.

Which of the error reactions occurs for the individual special function OBs is explained in the following sections.

Note
Calling a special function OB using the "JU OB 131/132/133" or "JC OB 131/132/133" operation does not have the same effect as a "genuine" block change, but functions as a STEP 5 operation without block operand. **No interrupts** are nested (with the default "interrupts at block boundaries").

6.1.3 Notes on the Special Functions OB 124, OB 125 and OB 254/255

- If, following the special function "delete block" (OB 124), you want to run the special function "generate block" (OB 125) or a comparable special function (OB 254/255) or the operation G DB/GX DX, a waiting time of 1 sec. must be allowed first. If this waiting time is not kept to, the standby CPU may stop and lead to an error entry "EDB error" in the error DB.

- Before you generate a logic block in your program, you must disable the updating of the standby with data by setting the H flag bit ARE .

- A block generated with OB 125 is considered as invalid by the self-test and not checked. Once you have written the required data to newly generated logic blocks (OB, PB, SB, FB or FX), you must call OB 125 again in the S5-155H to regenerate the checksum (enter the block length '0' when you call OB 125).

  After the checksum has been generated, the updating of the standby can be enabled again (reset H flag bit ARE).

Example: Sequence of operations for generating the checksum for the already generated PB 24.

```
:
:
:L    KB 0        Length = 0 (ID for checksum generation)
:L    KY 1.24     PB 24
:JU   OB 125
```

6.2 OB 121: Set/Read System Time

Function

With OB 121 you can set or read the system time (date and time). This function is compatible with the CPU 946R/947R.

Parameters

1. Data field

Four words in the word-oriented memory area.
With the "**set system time**" function the area **before** the OB121 call must be loaded with the time values to be set. The system program checks these values to ensure they are logically correct.
With the "**read system time**" function, OB121 enters the current time values in this area.

Structure of the data field

Bit no.	15 12	11 9	8 4	3 0
1st word	Sec. x 10	Sec. x 1	10th Sec.	100th Sec.
2ndword	Hour x 10	Hour x 1	Min. x 10	Min x 1
3rd word	Day x 10	Day x 1	Weekday	0
4th word	Year x 10	Year x 1	Month x 10	Month x 1

Note
The structure of the data field corresponds to the structure of system data RS 96 to RS 99 (current time).

Possible time values:

100th seconds: 0 to 9
10th seconds: 0 to 9

Seconds x 1: 0 to 9
Seconds x 10: 0 to 5

Minutes x 1: 0 to 9
Minutes x 10: 0 to 5

Hours x 1: 0 to 9
Hours x 10:
Bit no.
12 /13: 0 to 1 with 12 hour format
0 to 2 with 24 hour format
Bit no. 14: 0 = AM with 12 hour format
1 = PM " "
0 in 24 hour format
Bit no. 15: 0 = 12 hour format
1 = 24 hour format

Weekday:	0 to 6 for Mon to Sun
Days x 1:	0 to 9
Days x 10:	0 to 3
Month x 1:	0 to 9
Month x 10:	0 to 1
Year x 1:	0 to 9
Year x 10:	0 to 9

2. BR register (base address register)

Start address of a data field in an area of memory organized in words, from which the time values to be set are read or in which the current time values are to be stored. The BR register must be loaded with the address **before** the OB 121 call.

3. ACCU-1-L

Function no.,
Permitted values: 1 = set system time
2 = read system time

Result

After correct and error-free processing, the system program enters the value '0' in ACCU-1-L. After the "set" function, the system time is set to the values contained in the data field; after the "read" function, the data field contains the current time values.

Possible errors

The errors listed in the following table can occur. If one of these errors does occur, the system program enters the error ID shown in the table in ACCU-1-L.

Table 6-2 Error IDs of OB 121 in ACCU-1-L

ID	Meaning
F001H	Illegal function no.
F00FH	Multiple block call
F101H	Year specification illegal
F102H	Month specification illegal
F103H	Day specification illegal
F104H	Weekday specification illegal
F105H	Hour specification illegal
F106H	Minute specification illegal
F107H	Seconds specification illegal
F108H	100th ...10th seconds illegal
F109H	Hour format differs from setting in OB 151

Examples

Programming example for "set system time"

FB 13 is programmed for the "set system time" function. The new values are transferred in data block DB 10 (data word DW 0 to DW 3).

STEP 5 program:

```
   FB13
   NAME     :CLKWR
            :C    DB 10       Open DB 10
            :L    KH 1500     15 seconds (10th .. 100th sec. = 0!)
            :T    DW 0
            :L    KH 9555     24 hour mode, 15 hours 55 minutes
            :T    DW 1
            :L    KH 1010     the 10th, Tuesday
            :T    DW 2
            :L    KH 9308     1993, August
            :T    DW 3
            :MBR  EEC00       Load start address of the DB list
            :                 in the BR
            :LRW  +10         Load the start address of DB 10 in memory
            :                 (paragraph address) in ACCU 1
            :SLD  4           Absolute address of DB 10 (DW 0)
            :MAB              Load content of ACCU 1 into BR register
            :L    KB 1        Load function no. '1' in ACCU-1-L
            :JU   OB 121      Set system time
            :L    KB 0
            :><F              Scan error bits
            :JC   =ERRO       Jump to error handling
            :BEU
            :
   ERRO     :                 Error handling
            :
            :BE
```

Data block DB 10 contains the following information when OB 121 is called:

```
   0: KH = 1500;
   1: KH = 9555;
   2: KH = 1010;
   3: KH = 9308;
   4:
```

OB 121 transfers the required time parameters from DB 10 to the system data area RS 96 to RS 99.

Programming example for "read system time"

FB 14 is programmed for the "read system time" function. The current values should be stored in data block DB 11 (data word DW 0 to DW 3).

STEP 5 program:

```
   FB14
   NAME     :CLKRD
            :MBR EEC00      Load start address of the DB list
            :               in BR
            :LRW +11        Load start address of DB 11 in memory
            :               (paragraph address) in ACCU 1
            :L   KB  0
            :!=F            Check that OB 11 is loaded
            :JC  =NIVO      Jump to error handling
            :               if DB start address = 0
            :TAK
            :SLD 4          Absolute address of DB 11 (DW 0)
            :MAB            Load contents of ACCU 1 into BR register
            :L   KB 2       Load function no. '2' in ACCU-1-L
            :JU  OB 121     Read system time
            :BEU
            :
   NIVO     :               Error handling
            :
            :BE
```

Data block DB 11 contains the following information after OB 121 is called (example):

```
   0: KH = 2994;            29 sec., 940 millisec.
   1: KH = 9555;            24 hour format, 15 hours, 55 minutes
   2: KH = 1010;            10 days, weekday '1'(Tuesday), 0
   3: KH = 9308;            93 years, 8 months
   4:
```

It is Tuesday the 10th of August 1993, 15 hours, 57 minutes, 29 seconds and 940 milliseconds (9 tenths and 4 hundredths of a second).

6

6.3 OB 122: "Disable Interrupts" On/Off

A STEP 5 program can be interrupted at block boundaries or operation boundaries by programs at an execution level with a higher priority. The program execution levels with higher priority include the following:

- TIMED INTERRUPTS

  and

- PROCESS INTERRUPTS.

The time required to run the interrupted programs is expanded by the run time of the nested programs.

Function

Using OB 122, you can prevent interrupt servicing blocks being nested at one or more consecutive block or operation boundaries.

OB 122 influences the acceptance of interrupts:

- "Disable interrupts" on means the following:
  no interrupts will be registered from now on.

- "Disable interrupts" off means the following:
  all interrupts occurring will be registered from now on and the corresponding blocks will be nested and executed at the next operation or block boundary (depending on the mode set in DX 0).

Interrupts that have already been registered, are then no longer serviced if the mode "interruptability at block boundaries" is set in DX 0.

Parameters

ACCU-1-L

Function no.,
Permitted values: 1 = disable all interrupts
2 = enable all interrupts

Result

After correct and error-free processing, the system program enters the value '0' in ACCU-1-L.

> **Note**
> By calling OB 122, the RLO (undefined) is influenced. The BR register is not modified.
> To disable and enable processing interrupts, you can also use the STEP 5 operations IA and RA instead of OB 122. The blocking of interrupts is cleared at the next system checkpoint (refer to Chapter 10).

Possible errors

The errors listed in the following table can occur. If one of these error does occur, the system program writes the error ID shown in the table into ACCU-1-L.

Table 6-3 Error IDs of OB 122 in ACCU-1-L

ID	Meaning
F001H	Illegal function number

Example

```
Excerpt of a STEP 5 program in which all
interrupts are disabled using OB 122 immediately
before a critical program section following which
they are enabled again:

  :L   KB 1       Load function ID in ACCU-1-L
  :JU  OB 122     Disable all interrupts
  :             )
  :             )
  :             )
  :             } Critical program section
  :             )
  :             )
  :L   KB 2       Load function ID in ACCU-1-L
  :JU  OB 122     Enable all interrupts
  :
  :
```

6.4 OB 124: Delete STEP 5 Blocks

Function

With OB 24, you can delete any STEP 5 blocks (logic and data blocks) in the user memory. The deleted block is removed from the address list in DB 0.
The gap in memory resulting from deleting a block is used again when new blocks are loaded.

> **Note**
> If this function is used during START-UP, it must only be executed by the master.
> Please note the information in Section 6.1.3.

Parameters

1. ACCU-1-LH

Block **type** of the block to be deleted

2. ACCU-1-LL

Block **number** of the block to be deleted

Permitted block types and numbers:

ACCU-1-LH (block type)	ACCU-1-LL (block number)
1 = PB	0 to 255
2 = SB	0 to 255
3 = FB	0 to 255
4 = FX	0 to 255
5 = DB	3 to 255
6 = DX	3 to 255
7 = OB	1 to 39

Result

After correct and error-free processing, the system program sets the RLO to '0' and clears the condition codes CC 1 and CC 0.

> **Note**
> While the blocks are actually being deleted, user interrupts are disabled: no interrupts come through.
>
> By calling OB 124, the contents of ACCU 1 to ACCU 4 are modified. The BR register is retained.

Possible errors and warnings

If an error or warning occurs, the system program stops processing OB 124 and continues program execution at the next STEP 5 operation. It also sets the RLO to '1' and writes an ID to ACCU-1-LL (refer to Table 6-5).

If the function is aborted with a warning, it may be possible to achieve correct execution of OB 124 by re-calling the special function (possibly several times).

In the following case, OB 124 is aborted with a **warning**:

During the last 10 ms OB 124, OB 125, OB 254 or OB 255 has been called. (Only one call for these special functions is allowed within 10 ms. This avoids multiple calls for the OBs listed above blocking the interface to the PG so that it can no longer be processed.)

Condition code bits

After calling OB 124, you can check whether the special function has been executed correctly or was aborted with an "error" or "warning" using the result of logic operation and the condition code bits CC 1 and CC 0. The result can be evaluated with conditional jump operations.

Results bits

Table 6-4 Results bits of OB 124

RLO	CC 1	CC 0	Meaning	Scan
0	0	0	Special function was processed correctly	JC JZ
1	1	0	Processing of special function aborted with "warning"	JC JP JN
1	0	1	Processing of special function aborted with "error"	JC JM JN

IDs in ACCU-1-LL

In ACCU-1-LL, the system program stores IDs about the processing result, with which the cause of a warning or error is specified in more detail.

Bit no.	7	6	5	0
	W	E	Cause of error/warning	

The following group bits are fixed:

Bit no. 7 (W) = 1: warning
Bit no. 6 (E) = 1: error

Table 6-5 Result IDs of OB 124 in ACCU-1-LL

ID	Meaning
01H	Function was correctly processed
	Error:
45H	Block type not permitted
47H	Block does not exist
4DH	Online function COMPRESS MEMORY active
4FH	Function not possible at present, coupling standby active
	Warning:
8DH	Conflict with an online function (except for "compress memory")
8EH	10 ms waiting time not elapsed

Example

```
:L    KY 6,100   This sequence of operations deletes
:JU   OB 124     data block DX 100 in the user
:                memory
```

6.5 OB 125: Generate STEP 5 Blocks

Function

With OB 125, you can generate any STEP 5 blocks (logic and data blocks) in the user memory. Generating logic blocks should, however, be left to specialists.

The specified block is set up in the internal RAM with a block header and block body and entered in DB 0. The block body contains random data. For this reason, a newly generated block must first be written to before any useful data can be read out of it.

> **Note**
> If this function is used during START-UP, it must only be executed by the master.
> Please note the information in Section 6.1.3.

Parameters

1. ACCU-1-LH

Block **type** of the block to be generated

2. ACCU-1-LL

Block **number** of the block to be generated

Permitted block types and numbers

ACCU-1-LH (block type)	ACCU-1-LL (block number)
1 = PB	0 to 255
2 = SB	0 to 255
3 = FB	0 to 255
4 = FX	0 to 255
5 = DB	3 to 255
6 = DX	3 to 255
7 = OB	1 to 39

3. ACCU-2-L

Number of words (required block length without block header). The maximum assignable block length is 32762 data words. At present, approximately 2 K words can be edited with a PG.

If you have generated a logic block with OB 125 and have filled it with data, you can **generate a checksum** by calling OB 125 again with "**number of words**" = **0** in ACCU-2-L and the appropriate information in ACCU 1.

Result

After correct and error-free processing, the system program sets the RLO to '0' and clears the condition codes CC 1 and CC 0.

Note
While the blocks are actually being generated, user interrupts are disabled: no interrupts come through.

By calling OB 125, the contents of ACCU 1 to ACCU 4 are modified. The BR register is retained.

Possible errors and warnings

If an **error** occurs, the system program stops processing OB 125 and continues program execution at the next STEP 5 operation. It also sets the RLO to '1' and writes an ID to ACCU-1-LL (refer to Table 6-7).

If the function is aborted with a warning, it may be possible to achieve correct execution of OB 125 by re-calling the special function (possibly several times).

In the following case, OB 125 is aborted with a **warning**:

During the last 10 ms OB 124, OB 125, OB 254 or OB 255 has been called. (Only one call for these special functions is allowed within 10 ms. This avoids multiple calls for the OBs listed above blocking the interface to the PG so that it can no longer be processed.)

Condition code bits

After calling OB 125, you can check whether the special function has been executed correctly or was aborted with an "error" or "warning" using the result of logic operation and the condition code bits CC 1 and CC 0. The result can be evaluated with conditional jump operations.

Result bits

Table 6-6 Result bits of OB 125

RLO	CC 1	CC 0	Meaning	Scan
0	0	0	Special function was processed correctly	JC JZ
1	1	0	Processing of special function aborted with "warning"	JC JP JN
1	0	1	Processing of special function aborted with "error"	JC JM JN

IDs in ACCU-1LL

In ACCU-1-LL, the system program stores IDs about the processing result, with which the cause of a warning or error is specified in more detail.

Bit no.	7	6	5	0
	W	E	Cause of error/warning	

The following group bits are fixed:

Bit no. 7 (W) = 1: warning
Bit no. 6 (E) = 1: error

Table 6-7 Result IDs of OB 125 in ACCU-1-LL

ID	Meaning
01H	Function correctly processed
	Errors:
42H	Block already exists
43H	Not enough memory
44H	Block length not permitted
45H	Block type not permitted
4DH	Online function COMPRESS MEMORY active
4FH	Function not possible at present, since coupling standby active
	Warnings:
8DH	Conflict with an online function (except for "compress memory")
8EH	10 ms waiting time not yet elapsed

Example

```
:L    KF +2000   This sequence of operations
:L    KY 5,24    generates DB 24 with a length of
:JU   OB 125     2000 data words
:                (total length including header:
:                2005 words)
```

6.6 OB 131: Delete ACCUs 1, 2, 3 and 4

Function

By calling special function organization block OB 131 once, you can delete the contents of ACCUs 1 to 4 extremely simply. OB 131 overwrites all four registers with '0'.

Parameters

None

Result

The ACCUs 1 to 4 (each 32 bits) are deleted ('0').

Possible errors

None

6.7 OB 132/133: Roll-Up ACCU/Roll-Down ACCU

Function

OB 132 and OB 133 roll the ACCU contents up or down:

- OB 132 (roll up) moves the contents of ACCU 1 to ACCU 2, the contents of ACCU 2 to ACCU 3 etc.
- OB 133 (roll down) moves the contents of the ACCUs in the opposite direction: the contents of ACCU 1 to ACCU 4, ACCU 4 to ACCU 3 etc.

Parameters

None

Result

Figs. 6-1 and 6-2 show the ACCU contents **before** and **after** calling OB 132 and OB 133.

> **Note**
> With the STEP 5 operations ENT (extended operation set) and TAK (system operation) the ACCU contents can also be shifted (refer to Section 3.4.3).

Possible errors

None

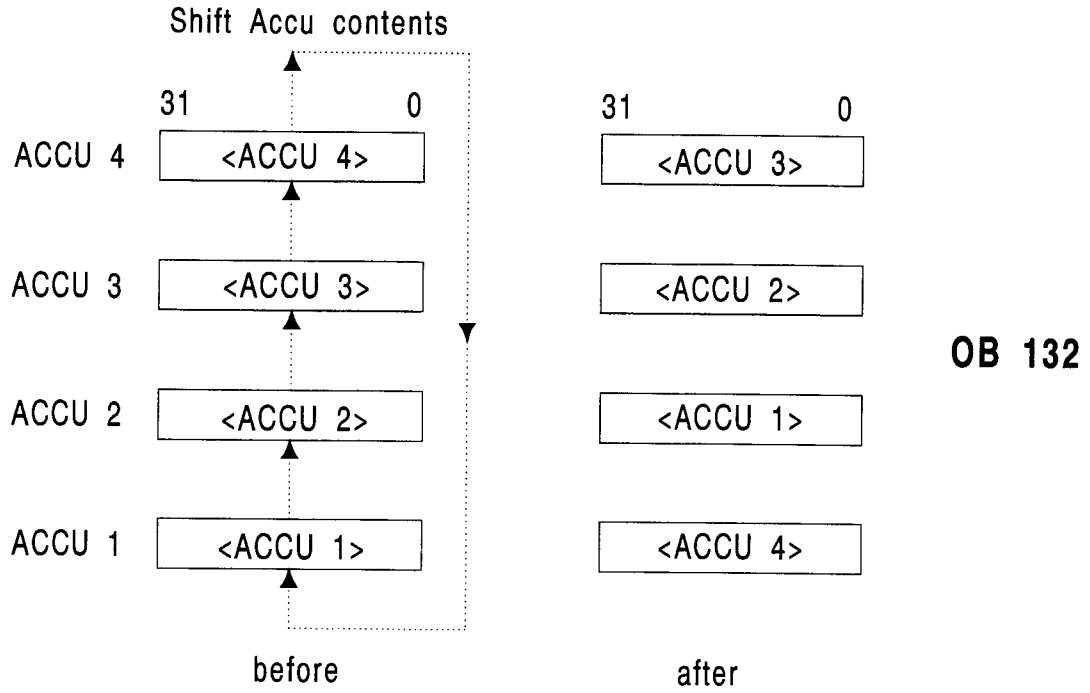


Fig. 6-1 Effect of the "roll-up" function

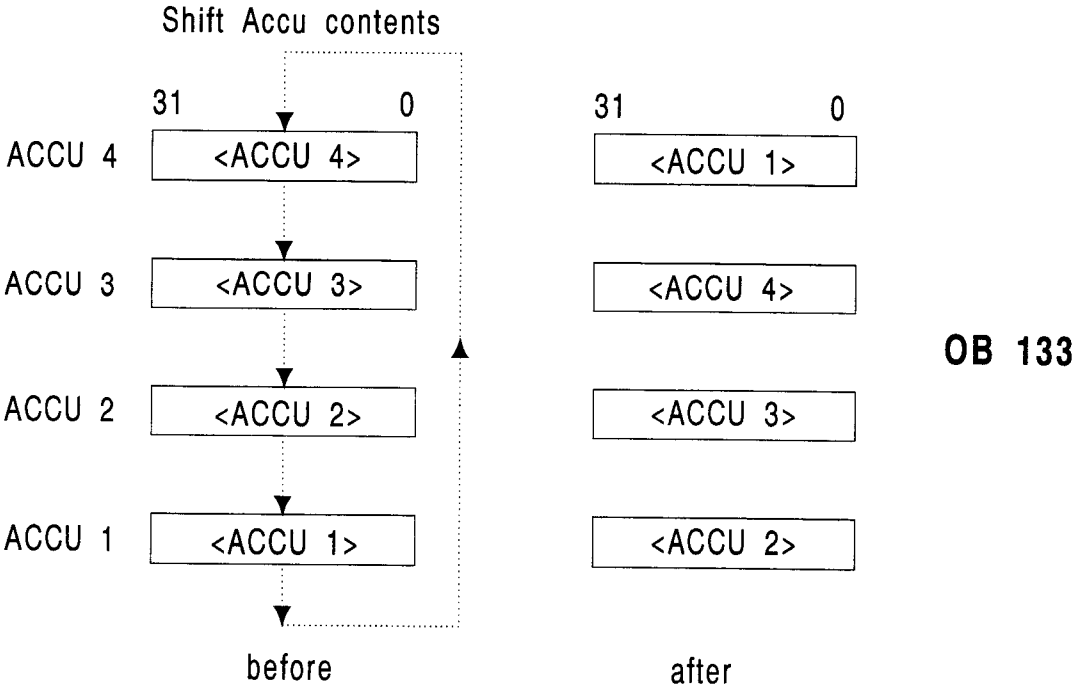


Fig. 6-2 Effect of the "roll-down" function

6.8 OB 141: "Disable Single Cyclic Timed Interrupts" On/Off

Using OB 141, you can prevent certain cyclic timed interrupt OBs (timed interrupts at **fixed intervals**) **from being called at one or more consecutive block or operation boundaries. For example, you can prevent an OB 10 (period 1) and an OB 11 (period 2) from being called in a particular program section that must not be interrupted. On the other hand, all remaining programmed timed interrupts are processed as usual.**

Function

OB 141 affects the reaction to cyclic timed interrupts.

"Disable single cyclic timed interrupts" **on** means that none of the specified cyclic timed interrupts are registered from this point onwards and the interrupts that were already registered (e.g. those waiting for a block boundary) are deleted. If a timed interrupt OB (for processing a timed interrupt with a fixed interval) has already been started, it is processed completely.

"Disable single cyclic timed interrupts" **off** means that all cyclic timed interrupts are registered again and are processed at the next block boundary .

Parameters

1. Control word

OB 141 records the timed interrupts to be disabled in a system-internal control word:



The bits of the control word have the following significance:

Bit no.	Interrupt
0 to 2	Reserved, these bits must be '0'
3 = '1' 4 = '1' 5 = '1' 6 = '1' 7 = '1' 8 = '1' 9 = '1' 10 = '1' 11 = '1'	Cyclic timed interrupts with fixed interval: period 1 (OB 10) period 2 (OB 11) period 3 (OB 12) period 4 (OB 13) period 5 (OB 14) period 6 (OB 15) period 7 (OB 16) period 8 (OB 17) period 9 (OB 18)
12 to 15	Reserved; these bits must be '0'

As long as a bit is set to '1' the corresponding interrupt is disabled.

2. ACCUs

2a) ACCU-2-L

Function no.,
Permitted values: 1, 2 or 3 where:

1: The contents of ACCU 1 are loaded in the control word.

2: All the bits marked '1' in the mask in ACCU 1 are set to '1' in the control word. The new control word is loaded in ACCU 1

3: All the bits marked '1' in the mask in ACCU 1 are set to '0' in the control word. The new control word is loaded in ACCU 1

2b) ACCU 1

New control word or mask depending on the required function.

Result

After correct and error-free processing the system program sets the RLO to '0'.
Calling OB 141 has the following results:

Funct. no. in ACCU-2-L	Contents of ACCU 1	
	before	**after**
1	control word	control word
2	mask	new control word
3	mask	new control word

Possible errors

If an error occurs, the system program sets the RLO to '1'.
The errors listed in the following table can occur. If an error occurs, the system program enters the error ID listed below in ACCU-1-L.

Table 6-8 Error IDs of OB 141 in ACCU-1-L

ID	Meaning
8D01H	illegal function no. in ACCU-2-L [1]
8D02H	one of the reserved bits in ACCU 1 is '1' [1]

[1] the incorrect value is located in ACCU-2-L

Scan control word

- The status of the control word can be scanned with the following program sequence:

  1. load function no. '2' or '3' in ACCU-2-L,

  2. load value '0' in ACCU 1,

  3. call OB 141,

  4. read out ACCU 1.

6.9 OB 142: "Delay All Interrupts" On/Off

A STEP 5 program can be interrupted at block boundaries by programs with a higher priority. The process interrupts and all timed interrupts belong to these higher priority program exection levels. The runtime of the interrupted program is extended by the runtime of the nested programs. Using OB 142, you can prevent the nesting of higher priority program execution levels at one or more consecutive block boundaries.

Function

OB 142 affects the servicing of interrupts:

"Delay interrupts" **on** means that all interrupts occurring are registered and pending interrupts remain registered. The registered interrupts are, however, initially not serviced. The operation or block boundaries for servicing interrupts are temporarily made ineffective. If an OB for process interrupt servicing or an OB for timed interrupt servicing has already started, this is processed completely.

"Delay interrupts" **off** means that all registered interrupts are processed at the next block or operation boundary.

> **Note**
> The time in which the interrupts are delayed must be shorter than **three times the value of the shortest timed interrupt period**. If this is not the case, a collision of timed interrupts occurs.

Parameters

1. Control word

OB 142 enters the interrupts to be delayed in a system-internal control word, as follows:



The bits in the control word have the following meaning:

Bit no.	Type of interrupt
0 = '1'	Cyclic timed interrupts, fixed period
1 = '1'	Real time interrupt
2 = '1'	Process interrupts
3 = '1'	Delayed interrupt
4 to 15	Reserved: these bits must be '0'

As long as a bit is set to '1', the corresponding interrupt is disabled.

2. ACCUs

2a) ACCU-2-L

Function no.,
Permitted values: 1, 2 or 3 where:

1: The contents of ACCU 1 are loaded in the control word.

2: All the bits marked '1' in the mask in ACCU 1 are set to '**1**' in the control word. The new control word is loaded in ACCU 1

3: All the bits marked '1' in the mask in ACCU 1 are set to '**0**' in the control word. The new control word is loaded in ACCU 1

2b) ACCU 1

New control word or mask depending on the required function.

Result

After correct and error-free processing the system program sets the RLO to '0'.
Calling OB 142 has the following results:

Funct. no. in ACCU-2-L	Contents of ACCU 1	
	before	after
1	control word	control word
2	mask	new control word
3	mask	new control word

Possible errors

If an error occurs, the system program sets the RLO to '1'.
The errors listed in the following table can occur. If an error occurs, the system program enters the error ID listed below in ACCU-1-L.

Table 6-9 Error IDs of OB 142 in ACCU-1-L

ID	Meaning
8E01H	Illegal function no. in ACCU-2-L [1]
8E02H	One of the reserved bits (no. 4 to 15) in ACCU 1 is '1' [1]
8EFFH	Incorrect mode (e.g. when the delayed interrupt is to be disabled and DX 0 contains the parameter "process interrupts via IB 0 = on")

[1] the incorrect value is located in ACCU-2-L

Scan control word

The status of the control word can be scanned with the following program sequence:

1. Load function no. '2' or '3' in ACCU-2-L,

2. Load value '0' in ACCU 1,

3. Call OB 142,

4. Read out ACCU 1.

6.10 OB 143: "Delay Single Cyclic Timed Interrupts" On/Off

Using OB 143, you can prevent certain cyclic timed interrupt OBs (timed interrupt OBs with a **fixed period**) from being called at one or more consecutive block or operation boundaries. For example, you can select a program section which cannot be interrupted by an OB 10 (period 1) and an OB 11 (period 2). On the other hand, all the remaining programmed timed interrupts are processed as usual.

Function

OB 143 affects the processing of cyclic timed interrupts:

"Delay single cyclic timed interrupts" **on** means that all interrupts are registered and pending timed interrupts remain registered. The timed interrupts specified in the control word are, however, not processed immediately. Temporarily, all the operation and block boundaries for processing these timed interrupts are made ineffective. If a timed interrupt OB (for processing a timed interrupt with fixed period) has already started, it is processed completely.

"Delay single cyclic timed interrupts" **off** means that all registered interrupts are serviced at the next block boundary.

Parameters

1. Control word

OB 143 enters the timed interrupts to be delayed in a system-internal control word:



The bits in the control word have the following meaning:

Bit no.	Interrupt
0 to 2	Reserved, these bits must be '0'
3 = '1' 4 = '1' 5 = '1' 6 = '1' 7 = '1' 8 = '1' 9 = '1'	Cyclic timed interrupts with fixed period period 1 (OB 10) period 2 (OB 11) period 3 (OB 12) period 4 (OB 13) period 5 (OB 14) period 6 (OB 15) period 7 (OB 16)
10 = '1' 11 = '1'	period 8 (OB 17) period 9 (OB 18)
12 to 15	Reserved, these bits must be '0'

As long as a bit is set to '1', the corresponding interrupt is disabled.

2. ACCUs

2a) ACCU-2-L

Function no.,
Permitted values: 1, 2 or 3 where:

1: The contents of ACCU 1 are loaded in the control word.

2: All the bits marked '1' in the mask in ACCU 1 are set to '1' in the control word. The new control word is loaded in ACCU 1

3: All the bits marked '1' in the mask in ACCU 1 are set to '0' in the control word. The new control word is loaded in ACCU 1

2b) ACCU 1

New control word or mask depending on the required function.

Result

After correct and error-free processing the system program sets the RLO to '0'.
Calling OB 143 has the following results:

Funct. no. in ACCU-2-L	Contents of ACCU 1	
	before	**after**
1	control word	control word
2	mask	new control word
3	mask	new control word

Possible errors

If an error occurs, the system program sets the RLO to '1'.
The errors listed in the following table can occur. If an error occurs, the system program enters the error ID listed below in ACCU-1-L.

Table 6-10 Error IDs of OB 143 in ACCU-1-L

ID	Meaning
8F01H	Illegal function no. in ACCU-2-L [1]
8F02H	One of the reserved bits in ACCU 1 is '1' [1]

1) the incorrect value is located in ACCU-2-L

Scan control word

- The status of the control word can be scanned with the following program sequence:

  1. load function no. '2' or '3' in ACCU-2-L,

  2. load value '0' in ACCU 1,

  3. call OB 143,

  4. read out ACCU 1.

6.11 OB 150: Set/Read System Time

Features of the system time

- The system time is backed up by the battery in the PLC rack. If the time is set, it therefore remains correct even following a power failure.
- The resolution is 10 ms for reading and 1 s for setting.
- Leap years are taken into account.
- Hours can be represented either using the 24 hour clock or the 12 hour clock "am" and "pm".
- The weekday is specified.
- Input and output is BCD coded.

Function

Using OB 150, you can set or read out the date and time of the CPU 948R in your user program. The date and time are known as the "system time".

> **Note**
> The system time is started (initially with a default value) after the CPU is plugged in.

Parameters

1. Data field for the time parameters

When **setting** the system time, OB 150 reads in the values to be set from a data field, when **reading** the time, OB 150 transfers the current time to the data field. You can set up this data field in a **data block** or in one of the two **flag areas** (F or S flags).

The data field consists of four words.

1a) Format of the data field for **setting** the system time.

Bit no.	15			12	11			8	7			4	3			0
word 1	Seconds								0							
word 2	Format	Hours							Minutes							
word 3	Day of month								Weekday			0				
word 4	Year								Month							

1b) Format of the data field when **reading** the system time

Bit no.	15			12	11			8	7			4	3			0
word 1	Seconds								1/100 seconds							
word 2	Format	Hours							Minutes							
word 3	Day of month								Weekday				0			
word 4	Year								Month							

The time parameters have the following meaning, range of values and representation:

Parameter	Permitted range of values	Value in
Seconds	0 to 59	BCD format
1/100 seconds	0 to 99 (when setting = 0)	
Minutes	0 to 59	
Hours	0 to 23 or 1 to 12, depending on the "format"	
Weekday	0 to 6 for Mon to Sun	
Day of month [1]	1 to 31	
	1 to 12	
Month	0 to 99	
Year		
Format	Format for the hour field with the following meaning: Bit 15 = 0: 12 hour format ("am" or "pm" selected in bit 14) Bit 15 = 1: 24 hour format (bit 14 = 0) Bit 14 = 0: "am" Bit 14 = 1: "pm"	–

1) After OB 150 is called, the specified value is checked logically for the correct data taking into account leap years.

Data field in the flag area

If you set up the data field in a flag area, you must take into account the following assignment of the data field words to the flag bytes. 'x' is the parameter "number of 1st data field word" which must be written to ACCU-1-L when OB 150 is called:

Bit no.	15 8	7 0
Data field word 1	Flag byte x	Flag byte x+1
Data field word 2	Flag byte x+2	Flag byte x+3
Data field word 3	Flag byte x+4	Flag byte x+5
Data field word 4	Flag byte x+6	Flag byte x+7

2. ACCUs

2a) ACCU-2-L

ACCU-2-L contains information about the required function and the data field used. It must have the following structure:

Bit no.	15			12	11			8	7			4	3			0
	Function no.				Address area type				Data block no.							

Parameters in ACCU-2-L

Function no.,
Permitted values: 1 = set system time
2 = read system time

Address area type,
Permitted values: 1 = DB data block
2 = DX data block
3 = F flag area
4 = S flag area

Data block no.,
Permitted values: 3 to 255 (only with address area type '1' or '2'; with address area type '3' or '4' irrelevant)

2b) ACCU-1-L

Number of the 1st data field word
Permitted values (depending on the address area type):

DB, DX:	0 to 2039
F flags:	0 to 248 (= no. of flag byte 'x ')
S flags	0 to 4088 (= no. of flag byte 'x ')

Result

After correct processing of OB 150, the RLO, the condition code bits OR, ERAB and OS = 0.

Possible errors

The errors listed in the following table may occur. If an error occurs, the system program sets the RLO to '1' and stores the error IDs listed in the table in ACCU 1.

Table 6-11 Error IDs of OB 150 in ACCU-1-L

ID	Meaning
9601H	Data block not loaded
960FH	Multiple call for the block
9611H	Illegal function no.
9612H	Address area type illegal
9613H	Data block number illegal
9614H	"Number of first data field word" illegal
9615H	Data block length < 4 words
9621H	Year specification in the data field illegal
9622H	Month specification in the data field illegal
9623H	Day of month specification in the data field illegal
9624H	Weekday specification in the data field illegal
9625H	Hour specification in the data field illegal
9626H	Minute specification in the data field illegal
9627H	Second specification in the data field illegal
9628H	1/100 seconds in the data field not equal to 0
9629H	Hour format not equal to setting in OB 151

Note
If the parameters are incorrectly assigned for the "set system time" function and if the time has already been set correctly at least once, the error IDs listed are transferred; the previously set system time, however, continues to run.

Examples

```
"Set system time":

You want to set the system time to the following values:

   "Thurs, 24.10.1993, 11:30 hours 0 seconds, 24 hour clock"

The time parameters are stored in data block DB 10 from data word DW 0
onwards.


   DB 10
        0: KH =  0 0 0 0    left byte = seconds (BCD), right byte = 0

        1: KH =  9 1 3 0    91 = Format (=80H) + hour (=11 BCD)
                            30 = Minutes (BCD)
        2: KH =  2 4 3 0    24 = Day of month (BCD)
                            30 = Weekday (3 = Thursday) + bit 0 to 3 = 0
        3: KH =  9 3 1 0    93 = Year (BCD)
                            10 = Month (BCD)

                                        Continued on following page
```

"Set system time" *(continued)*

STEP 5 operations in OB 1 for calling OB 150:

```
        :
        :
        :L    KH  1 1 0 A  Values for ACCU-2-L:
        :                  DB no. = 10
        :                  Address area type = 1 for "data field in DB"
        :                  Function no. = 1 for "set"
        :L    KF +0        ACCU-1-L:
        :                  No. of 1st data field word = 0
        :JU   OB 150       Call OB 150
        :
```

"Read out system time":

You want the current system time to be written to data block DB 10 from data word DW 4 onwards. You must therefore call OB 150 with the following parameters:

```
        :
        :
        :L    KH  2 1 0 A                Values for ACCU-2-L:
        :                  DB no. = 10
        :                  Address area type = 1 for "data field in DB"
        :                  Function no. = 2 for "read"
        :L    KF +4        ACCU-1-L:
        :                  No. of 1st data field word = 4
        :JU   OB 150       Call OB 150
        :C    DB 10        Open DB 10
        :                  Evaluate DB 10
```

After OB 150 is called, the current system time is written to data block DB 10 in the following form ("Thurs, 24.10.93, 11:30 hours 20 seconds, 13/100 of a second, 24 hour clock"):

```
  DW 4:   KH =  2 0 1 3    Seconds = 20 (BCD)
                           1/100 seconds = 13 (BCD)
  DW 5:   KH =  9 1 3 0    Format = 24 hour (bit 15 = 1, bit 14 = 0),
                           Hour = 11(BCD), minutes = 30 (BCD)
  DW 6:   KH =  2 4 3 0    Day of month = 24 (BCD)
                           Weekday = 3 = Thursday
  DW 7:   KH =  9 3 1 0    Year = 93 (BCD)
                           Month = 10 (BCD)
```

6.12 OB 151: Set/Read Time for Real Time Interrupt

Function

By calling OB 151 you can do the following:

- cause the CPU 948R to activate the real time interrupt ("timed job" - OB 9, refer to Section 4.5.3) at a selected time:
  - every minute
  - every hour
  - every day
  - every week
  - every month
  - every year
  - once,
- read out the current status of a timed job,
- cancel a previously generated timed job.

OB 151 can be called in the START-UP and RUN modes. A generated real time interrupt is retained when a WARM RESTART (automatic or manual) is carried out. A COLD RESTART deletes an existing timed job.

If you generate a new timed job, an existing timed job is automatically cancelled. This means that only **one** real time interrupt can be active.

Parameters

1. Data field for job parameters

When **generating** or **cancelling** a timed job, OB 151 takes the required job parameters from a data field. When **reading out** the current status of the job management, OB 151 transfers the current job parameters to a data field.

You can set up this data field in a **data block** or in one of the two **flag areas** (F or S flags).

The data field consists of four words and has the following format when generating and reading out a timed job:

Bit no.	15			12	11			8	7			4	3			0
Word 1	Seconds								0							
Word 2	Format		Hours						Minutes							
Word 3	Day of month								Weekday				Job type			
Word 4	Year								Month							

The parameters have the following meaning, range of values and representation:

Parameter	Permitted range of values	Value in
Job type	0 to 7 where: 0: cancel job or no job active 1: every minute 2: every hour 3: every day 4: every week 5: every month 6: every year 7: once	BCD format
Seconds 1/100 seconds Minutes Hours Weekday Day of month [1] Month Year	0 to59 0 to 99 (when setting = 0) 0 to 59 0 to 23 or 1 to 12, depending on the format 0 to 6 for Mon to Sun 1 to 31 1 to 12 0 to 99	BCD format
Format [2]	Format for the hour field with the following meaning: Bit 15 = 0: 12 hour clock Bit 15 = 1: 24 hour clock (bit 14 = 0) Bit 14 = 0: "am" Bit 14 = 1: "pm"	–

1) After calling OB 151, the specified value is checked logically that the date is correct taking into account leap years.
2) For the meaning of "am" and "pm": refer to OB 150 in the previous section: "format" must match the form specified when setting the system time with OB 150.

Data field in the flag area

If you set up the data field in a flag area, you must take into account the following assignment of the data field words to the flag bytes. "x" is the parameter 'number of the 1st data field word' which must be written to ACCU-1-L when OB 151 is called.

Bit no.	15 8	7 0
Data field word 1	Flag byte x	Flag byte x+1
Data field word 2	Flag byte x+2	Flag byte x+3
Data field word 3	Flag byte x+4	Flag byte x+5
Data field word 4	Flag byte x+6	Flag byte x+7

2. ACCUs

2a) ACCU-2-L

ACCU-2-L contains information about the required function and the data field used. It must have the following structure:

Bit no.	15			12	11			8	7			4	3			0
	Function no.				Address area type				Data block no.							

Parameters in ACCU-2-L

Function no.,
Permitted values: 1 = generate job
2 = read job status

Address area type,
Permitted values: 1 = DB data block
2 = DX data block
3 = F flag area
4 = S flag area

Data block no.,
Permitted values: 3 to 255 (only with address area type '1' or '2'; with address area type '3' or '4' irrelevant)

2b) ACCU-1-L

No. of the 1st data field word,
Permitted values (depending on the address area type):

DB, DX:	0 to 2039
F flags:	0 to 248 (= no. of flag byte 'x ')
S flags	0 to 4088 (= no. of flag byte 'x ')

Note
There is no point in generating a timed job cyclically (e.g. with an unconditional OB 151 call with function number '1' in OB 1).

Result

After correct processing of OB 151, the RLO, the condition code bits OR, ERAB and OS = 0.

Note
If, when reading out the timed job, the job type is '**0**' and all the remaining parameters are '**F**' or '**FF**' (hex) in the data field, no timed job is active.
This status can occur in the following situations:

a) there was a COLD RESTART without a timed job being generated

b) when a "one-time" timed job was due

or

c) when you have cancelled a job.

Possible errors

The errors listed in the following table can occur. If one of these errors occurs, the system program sets the RLO to '1' and writes the error IDs listed in the table to ACCU 1.

Table 6-12 Error IDs of OB 151 in ACCU-1-L

ID	Meaning
9701H	Data block not loaded
970FH	Multiple call for the block
9710H	Wrong mode ("process interrupts via IB 0 = on")
9711H	Illegal function no.
9712H	Address area type illegal
9713H	Data block no. illegal
9714H	"Number of the 1st data field word" illegal
9715H	Data block length less than 4 words
9721H	Year specification in the data field illegal
9722H	Month specification in the data field illegal
9723H	Day of month specification in the data field illegal
9724H	Weekday specification in the data field illegal
9725H	Hour specification in the data field illegal
9726H	Minute specification in the data field illegal
9727H	Second specification in the data field illegal
9728H	1/100 seconds in the data field not equal to 0
9729H	Hour format not equal to setting in OB 121/OB 150
972AH	Job type illegal

Note
If incorrect parameters are assigned **and** a valid timed job was previously generated, the error IDs listed above are transferred; **the previously generated timed job, however, remains active.**

Points to note with the time parameters

Regardless of when a real time interrupt (timed job) is to be triggered, the individual time parameters must be specified in certain combinations. Depending on the selected time for the real time interrupt, certain parameters must be specified, while others are not evaluated by the system program.

The following table shows which time parameters must be specified for which timed job (XXX = must be specified, --- = irrelevant).

Table 6-13 Assignment of "timed job - time parameters"

Interval	Seconds	Minutes	Hours	Weekday	Day of month	Month	Year
every minute	XXX	---	---	---	---	---	---
every hour	XXX	XXX	---	---	---	---	---
every day	XXX	XXX	XXX	---	---	---	---
every week	XXX	XXX	XXX	XXX	---	---	---
every month	XXX	XXX	XXX	---	XXX	---	---
every year	XXX	XXX	XXX	---	XXX	XXX	---
once	XXX	XXX	XXX	---	XXX	XXX	XXX

When **reading out** the time parameters, the irrelevant parameters are assigned the value FFH.

Special situations

- If the "29th of February" is selected with the job type "every year" (= 6), this means that OB 9 is only called every leap year.
- If the value "29", "30" or "31" is selected with the job type "every month" (= 5), OB 9 is only called in the months which have these dates.

Examples

Various timed jobs (in 24 hour format):

1. "Job at 29th second of every minute"
   (12:44:29, 12:45:29 etc.):

You must specify:	Job type	= 1	(Function no. in ACCU-2-L = 1)
	Seconds	= 29	

2. "Job every hour at xx:14:15":

You must specify:	Job type	= 2	(Function no. in ACCU-2-L = 1)
	Seconds	= 15	
	Minutes	= 14	

3. "Job daily at 5:32:47":

You must specify:	Job type	= 3	(Function no. in ACCU-2-L = 1)
	Seconds	= 47	
	Minutes	= 32	
	Format/hour	= 85	

4. "Job weekly, Tuesdays at 10:50:00":

You must specify:	Job type	= 4	(Function no. in ACCU-2-L = 1)
	Seconds	= 00	
	Minutes	= 50	
	Format/hour	= 90	
	Weekday	= 01	

5. "Job monthly, on the 14th at 7:30:15":

You must specify:	Job type	= 5	(Function no. in ACCU-2-L = 1)
	Seconds	= 15	
	Minutes	= 30	
	Format/hour	= 87	
	Day of month	= 14	

6. "Job yearly, on the 1st of May at 00:01:45":

You must specify:	Job type	= 6	(Function no. in ACCU-2-L = 1)
	Seconds	= 45	
	Minutes	= 01	
	Format/hour	= 80	
	Day of month	= 01	
	Month	= 05	

Continued on the following page

Various timed jobs (in 24 hour format), *continued* :

7. "Job once on the 31.12.1999 at 23:55:00":

You must specify:	Job type	=	7 (Function no. in ACCU-2-L = 1)
	Seconds	=	00
	Minutes	=	55
	Format/hour	=	A3
	Day of month	=	31
	Month	=	12
	Year	=	99

8. "Cancel job":

You must specify:	Job type	=	0 (Function no. in ACCU-2-L = 1)

9. "Read out time job":

You must specify: Function no. in ACCU-2-L = 2

If no job is active, you obtain the following result in the data field:

Data field word 0:	FFFF H
Data field word 1:	FFFF H
Data field word 2:	FFF0 H
Data field word 3:	FFFF H

6

6.13 OB 153: Set/Read Time for Delayed Interrupt

Using OB 153, you can transfer so-called "delay jobs" to the system program. After a specified delay time "a delayed interrupt" is then processed (refer to OB 6, Section 4.5.3).

Function

By calling OB 153, you can do the following:

- define and start a delay time,
- stop an activated delay time (cancel delay job),
- read how long the delay time still has to run.

A delay job can be activated in the START UP and RUN modes via OB 153.

Life of a delay job

The delayed interrupt triggered by a delay job is only activated by the system program in the **RUN** mode (OB 6 call).
Jobs which become due in a mode other than RUN are discarded by the system program **without any message**.
A currently active (but not yet due) job is also discarded if the CPU changes to the STOP mode or if the power is switched off.

Parameters

ACCUs

a) ACCU-2-L

ACCU-2-L only needs to be supplied with the function number '**1**' ("define delay time") when OB 153 is called, as follows:

Delay time in milliseconds (max. 65535)

Permitted values: 0005H to FFFFH

b) ACCU-1-L

Function no.

Permitted values: 1 = define and start delay time
2 = stop delay time (= cancel job)
3 = read remaining delay time

Note
If a previously defined delay time is not yet elapsed when a further delay time is defined, the previously defined time is lost and the new delay time started.

Result

After correct processing of OB 153, the RLO, the condition code bits OR, ERAB and OS = 0.

When OB 153 is called with the function no. '2' or '3', ACCU-1-L contains the remaining time to run in milliseconds.

If no delay job is active when OB 153 is called with function no. '2' or '3', ACCU-1-L contains the value '0'.

Possible errors

The errors listed in the following table can occur. If one of the errors occurs, the system program sets the RLO to '1' and writes the error IDs listed in the table to ACCU 1.

Table 6-14 Error IDs of OB 153

ID	Meaning
990FH	Multiple call for the block
9910H	Wrong mode ("process interrupt via IB 0 = on")
9911H	Illegal function number
9921H	Delay time illegal

Examples

Define and start delay time:

When an AUTOMATIC WARM RESTART is performed, after 5 seconds a certain STEP 5 operation sequence must be run through once. To do this, the delay time is defined and started in start-up organization block OB 22.

The STEP 5 operations in OB 22 for calling OB 153:

```
      :
      :
      :L    KF  +5000     Value for ACCU-2-L: 5000 ms
      :L    KF  +1        Value for ACCU-1-L: function no. = 1 for
      :                   "define and start delay time"
      :JU   OB 153        Call OB 153
      :
```

Stop delay time (cancel job)

STEP 5 operations for calling OB 153:

```
        :
        :
        :L    KF  +2        Value for ACCU-1-L: function no. = 2 for
        :                   "stop delay time"
        :JU   OB 153        Call OB 153
        :
        :
```

Read out remaining time of a delay job:

STEP 5 operations for calling OB 153:

```
        :
        :
        :L    KF  +3        Value for ACCU-1-L: function no. = 3 for
        :                   "read out remaining time"
        :JU   OB 153        Call OB 153
        :
        :                   ACCU-1-L contains the time the delay job still
        :                   has to run.
        :
```

6.14 OB 222: Restart Cycle Monitoring Time

The special function OB 222 causes the cycle monitoring time to be restarted, i.e. the timer for monitoring is started from the beginning. By calling this special function, the maximum permitted cycle time for the current cycle is extended by the value selected at the time of the call.

Parameters none

Possible errors none

6.15 OB 254/255: Copy/Duplicate Data Blocks

With the special functions OB 254/255, you copy individual data blocks from a memory card to the user memory or duplicate individual data blocks within the user memory.
The special function OB 254 and OB 255 work identically, with OB 254 exclusively for DX data blocks and OB 255 for DB data blocks.

Note

- During the copying or duplication, the user interrupts are blocked. No timed interrupts or process interrupts are accepted.
- If one of the functions is used in the START-UP, it must only be executed by the master.
- Note the information in Section 6.1.3.

Application

Copying data blocks from the memory card or duplicating data blocks in the user memory and assigning a new block number.

Copying

Conditions

By using the "**copy**" function of the two special function OBs (OB 254 or OB 255 call), remember the following conditions:

- **The memory card must be plugged in before the OVERALL RESET and must not be removed afterwards**
- The destination data block must not yet exist
- The online function "compress memory" must not be active

Function

A data block is copied from the memory card to the user memory and retains its original block number. The start address is entered in the address list in DB 0.

Parameters

1. ACCU-1-LL

Number of the block to be copied.

The following block numbers are possible:

Block type	Block number
DB (OB 255) DX (OB 254)	3 to 255 3 to 255

2. ACCU-1-LH

ACCU-1-LH must be zero.

Duplicating

Function

A data block is duplicated within the user memory and it is assigned a different block number. The start address of the new data block is entered in the address list in DB 0. The start address of the old block is retained, i.e. the original data block remains valid.

The start address is entered in DB 0 only after the transfer is completed and all the IDs are correctly entered in the block header. The duplicated block is therefore only declared valid or existent by the system program after it has been completely transferred.

Parameters

1. ACCU-1-LL

Number of the block to be duplicated (source).

2. ACCU-1-LH

Number of the new block (destination).

The following block numbers are possible:

Block type	Block number
DB (OB 255) DX (OB 254)	3 to 255 3 to 255

Result following copying and duplicating

After the function has been executed correctly and error-free, the system program sets the RLO to '0' and clears the condition code bits CC 1 and CC 0.

Calling OB 254/255 changes the contents of ACCU 1 to ACCU 4. The BR register is retained.

Possible errors and warnings when copying and duplicating

If an error or warning occurs, the system program stops processing OB 254/255 and continues program execution at the next STEP 5 operation. It also sets the RLO to '1' and writes an ID to ACCU-1-LL (refer to Table 6-18).

If the function is aborted due to a warning, it may be possible to run OB 254/255 correctly by calling the special function again (if necessary, repeated several times).

In the following situation, OB 254/255 is aborted with a **warning**:

an OB 124, OB 125, OB 254 or OB 255 has been called during the last 10 ms. (During a period of 10 ms, however, only one special function OB call is permitted. This prevents multiple calls for the OBs listed above preventing the interface to the PG from being processed.)

Condition code bits following copying and duplicating

After OB 254/255 is called, you can see whether the special function has been carried out correctly or whether it was stopped by an "error" or "warning" based on the result of logic operation and the condition code bits CC 1 and CC 0. The result can be evaluated by conditional jump operations.

Result codes

Table 6-15 Result codes for OB 254/255

RLO	CC 1	CC 0	Meaning	Scan
0	0	0	Special function correctly processed	JB JZ
1	1	0	Special function aborted with "warning"	JB JP JN
1	0	1	Special function aborted with "error"	JB JM JN

IDs in ACCU-1-LL

The system program sets IDs in ACCU-1-LL which specify the causes of a warning or error in more detail.

Bit no.	7	6	5	0
	W	E	Cause of error/warning	

The following group codes apply:

Bit no. 7 (W) = 1: warning
Bit no. 6 (E) = 1: error

Table 6-16 Result IDs for OB 254/255 in ACCU-1-L

ID	Meaning
01H	Function correctly executed
	Error:
41H	Block header on memory card invalid
43H	Not enough memory space
48H	Source data block does not exist
4AH	Block number or type illegal/source DB
4BH	Block number or type illegal/destination DB
4CH	Destination data block already exists in the user memory
4DH	Online function COMPRESS MEMORY active
4EH	No memory card plugged in
4FH	Function not possible at present, since coupling standby active
	Warnings:
8DH	Conflict with an online function except "compress memory"
8EH	10 ms waiting time not yet elapsed

Examples

```
1. "Copy":

:L   KY 0,120   This sequence of operations copies
:JU  OB 254     the data block DX 120 from the
:               memory card to the user memory
:

2. "Duplicate":

:L   KY 80,85   This sequence of operations
:JU  OB 255     duplicates data block DB 85; the
:               new data block has the number 80.
:               The contents of DB 80 and DB 85
:               are identical.
:
```

Extended Data Block DX 0

7

Contents of Chapter 7

7.1 Application ..... 7 - 4

7.2 Parameter Assignment Using the PG Screen Form ..... 7 - 5

7.3 Meaning of Some Parameters for Running the Program ..... 7 - 7

7.3.1 Updating the Timers ..... 7 - 7
7.3.2 Setting the Cycle Monitoring Times ..... 7 - 7
7.3.3 Level Priorities ..... 7 - 7

Extended Data Block DX 0

7

The following chapter explains how to use the data block DX 0. You will find information about assigning parameters using a screen form and the meaning of the various DX 0 parameters.

7.1 Application

You can adapt certain system program functions to meet your own requirements by selecting alternative settings in the DX 0 screen form compared with the standard defaults .

The defaults of the system program are automatically set during each COLD RESTART and DX 0 is then evaluated. If you have not assigned parameters for DX 0 and loaded it, the defaults remain valid; otherwise the defaults you have selected in DX 0 become the valid settings.

You can make settings in DX 0 if the PG software STEP 5/ST, Version 6.3 or STEP 5/MT, Version 6.0 plus the corresponding "Delta diskette CPU 948" is installed on your PG.

> **Note**
> The settings or modifications made in DX 0 only become effective following a COLD RESTART.
> If a **modified DX 0** is read during a COLD RESTART, the **unmodified** parameter assignments are **retained**.

Differences compared with the CPU 946R/947R

Compared with DX 0 parameter assignment for the CPU 946R/947R, there are various differences when assigning DX 0 parameters for the CPU 948R, as follows:

- Modes:
  There is no longer a 150U mode, instead of this, there is now "**interruptability at block boundaries**".

- Additional timed interrupts:
  If you switch off the mode "process interrupts via IB 0", the additional timed interrupts the **delayed interrupt (OB 6)** and the **real time interrupt (OB 9)** are available.

7.2 Parameter Assignment using the PG Screen Form

Screen forms are available for assigning parameters in DX 0 for the CPU 948R. The PG software automatically generates data block DX 0 according to the default parameters (in bold face) and parameters you have specified. To assign parameters to DX 0, two screen forms are required.

How to select and complete the PG screen forms is explained in your STEP 5 manual.

Layout of the screen forms

Two screen forms are required for completing parameter assignment to DX 0:

The first screen form (Fig. 7-1) contains the parameter groups

- Interruptability
- Restart after power up,
- Warm restart procedure
- Number of timer cells
- Cycle time monitoring
- Synchronize multiprocessor restart
- Block transfer of the IPC flags

DX 0 - param. ass. (S5-155H CPU 948R)

Interruptability:	**at block bounds.** [1]
Restart after power up:	**1** (1 = warm restart [2] 2 = cold restart 3 = manual start)
Warm restart procedure:	**1** (1 = warm restart [3] 2 = cold restart with memory)
Number of timer cells:	**256** (0...256)
Cycle time monitoring (x 10 ms):	**20** (1...255)
Synchronize multiprocessor restart:	**YES**
Block transfer of the IPC flags:	**NO**

F1	F2	F3 SELECT	F4	F5	F6 CONTINUE	F7	F8

1) with older PG-software versions the following is displayed:
Mode 150U for "interruptability at block boundaries"
Mode 155U for "interruptability at operation boundaries"

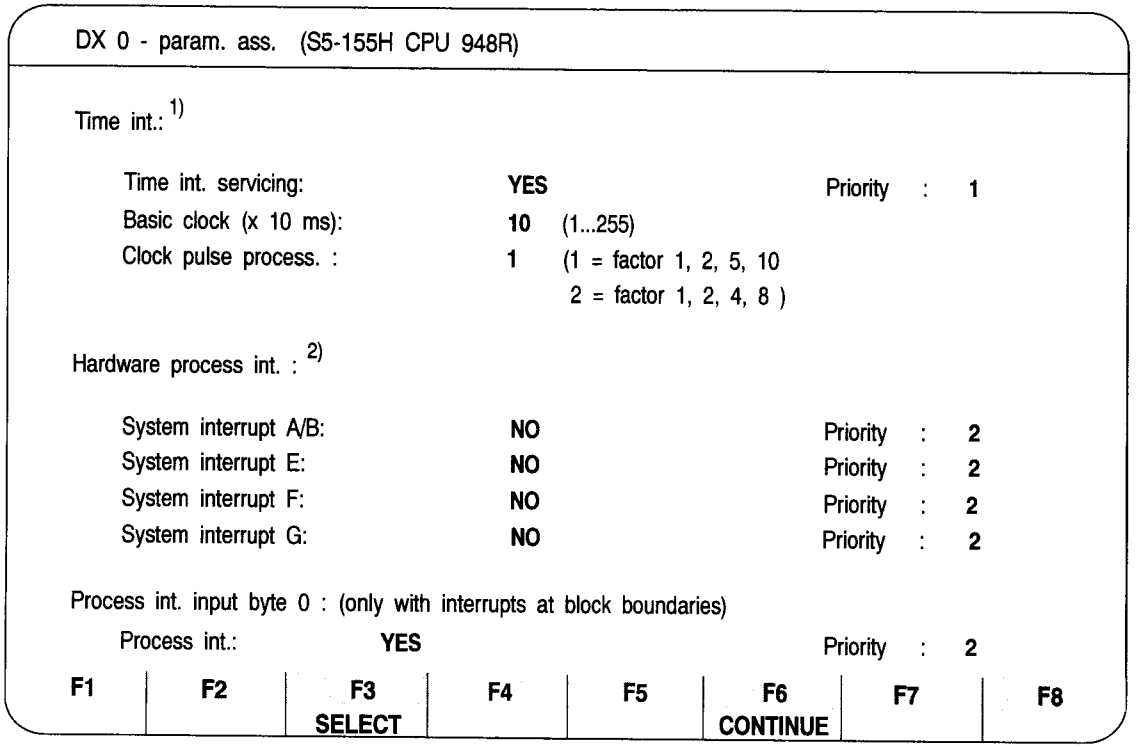
2) executed with "cold restart" with memory

3) only option '2' can be selected

Fig. 7-1 PG screen form for assigning parameters to DX 0 / Part 1

If you move on to the second screen form (Fig. 7-2) you will find the following parameters:

- Time interrupts
- Hardware process interrupts,
- Process interrupts input byte 0.



1) The delayed interrupt and real time interrupt must be activated extra by switching off process interrupts

Fig. 7-2 PG screen form for assigning parameters to DX 0 / Part 2

7.3 Meaning of Some Parameters for Running the Program

7.3.1 Updating the Timers

The system program processes the timers as shown below:

- As standard, the timers T 0 to T 255 are updated.
- If you enter the value "0" in DX 0, **no** timers are updated, even if they are included in the program. There is then also no error message output.
- Updating is as follows:

Entry	'0'	'1'	'2'	'3'	'4'	....
Updating	none	T0 to T1	T0 to T2	T0 to T3	T0 to T4	....

7.3.2 Setting the Cycle Monitoring Times

You can set the cycle monitoring time with the OB 31 or DX 0 parameter.
If the cycle monitoring time is set with both OB 31 and DX 0, the system program uses the setting from OB 31 because it evaluates DX 0 earlier. For this reason, you should not use both possibilities at the same time. We recommend setting the cycle monitoring time in DX 0.
Since PG functions extend the cycle time, you should add a further 30% to the cycle time setting.

7.3.3 Level Priorities

You can specify different priorities for the program execution levels. This is achieved either using the default or by specifying DX 0 parameters.

Priorities when "process interrupts via IB 0 = on" is selected (PROCESS INTERRUPTS level)

These priorities have the following default values in DX 0:

- Timed interrupts: level priority 1 (higher priority)
- Process interrupts via input byte IB 0: level priority 2 (lower priority)

You can swap over the priorities in DX 0.

Memory Assignment and Memory Organization

8

Contents of Chapter 8

8.1	Structure of the Memory Area	8 - 4
8.2	Memory Assignment in the CPU 948R	8 - 5
8.2.1	Memory Assignment for the System RAM	8 - 6
8.2.2	Memory Assignment for the I/O	8 - 8
8.3	User Memory Organization in the CPU 948R	8 - 10
8.3.1	Block Headers in User Memory	8 - 12
8.3.2	Block Address List in Data Block DB 0	8 - 13
8.3.3	RI/RJ Area	8 - 14
8.3.4	RS/RT Area	8 - 15
8.3.5	Bit Assignment of the System Data Words	8 - 18
8.3.6	Addressable System Data Area	8 - 33

Memory Assignment and Memory Organization

8

You can use this chapter as a reference section to check on the organization of the CPU 948R memory. The chapter also includes important information contained in some of the system data words.

8.1 Structure of the Memory Area

The memory of the CPU 948R is essentially divided into the following areas:

Table 8-1 Structure of the memory area

Memory area		Data width	Location
User memory for: OBs, FBs, FXs, PBs, SBs, DBs, DXs		16 bits	CPU internal
Serial communications interface area:	RI, RJ	16 bits	
System area:	RS, RT	16 bits	
Timers:	T	16 bits	
Counters:	C	16 bits	
Flags:	F	8 bits	
Flags:	S	8 bits [1]	
Process image (PI) inputs/ outputs:	PII, PIQ	8 bits	
Peripheral (I/O) area, divided into:			On the S5 bus
	"P" peripherals	8 bits	
	"O" peripherals	8 bits	
	Interprocessor communication flags	8 bits	
	Coordinator (COR) (semaphore, ...)	8 bits	
	Dual-port RAM pages (CP, IP, COR 923C)	8/16 bits	
	IM3/IM4 area	8 bits	
	Hardware registers	8/16 bits	

[1] S flags occupy 8 bits in the 16-bit area. The high byte is undefined.

The next section lists the addresses of the memory areas shown.

Note
When using STEP 5, you should not access a memory register within an operand area (e.g., flags) directly via the absolute address of the memory register. This can result in undesirable operating statuses. Access it only relative to the base address of its operand area.

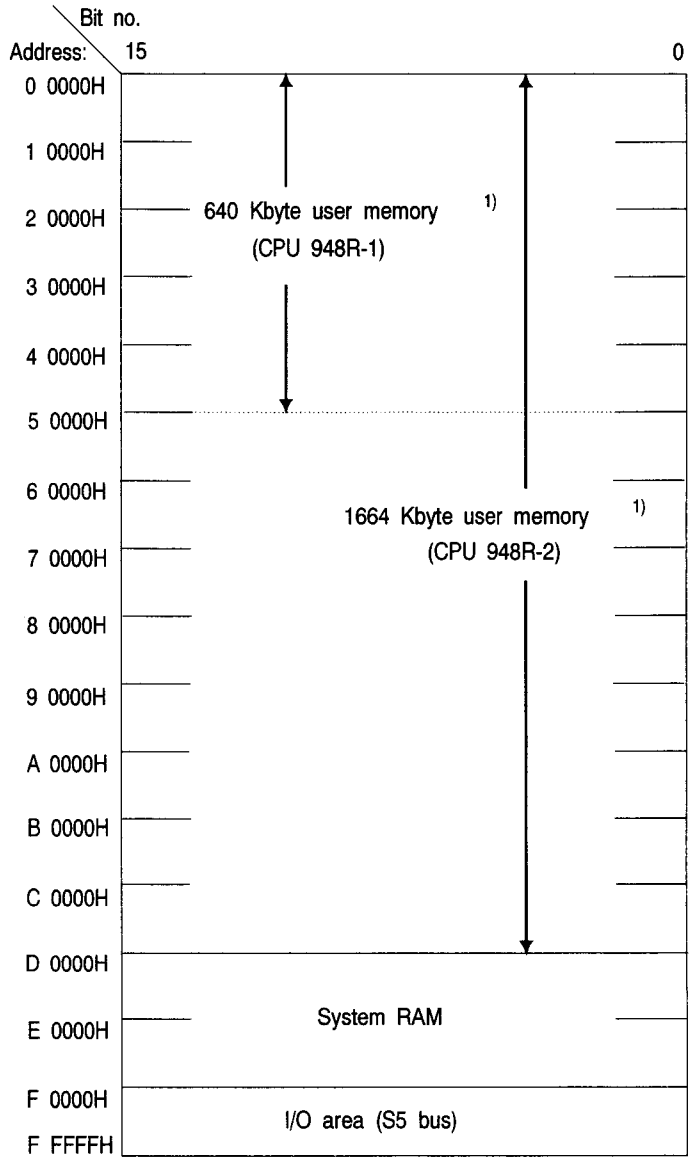
Direct access to the areas I, Q and F result in 'FFH' in the high byte and the data in the low byte. For direct access to S flags, the **high byte is undefined**!

8.2 Memory Assignment in the CPU 948R

With the CPU 948R, there are two possible user memories (RAM) available:

- the CPU 948R-1 with 640 Kbytes of user memory
- the CPU 948R-2 with 1664 Kbytes of user memory.

Fig. 8-1 illustrates the distribution of the address area of the CPU 948R and the location of the user memory versions.



1) The last 20 words of the user memory cannot be used.

Fig. 8-1 Memory assignment in CPU 948R/overview

8

8.2.1 Memory Assignment for the System RAM

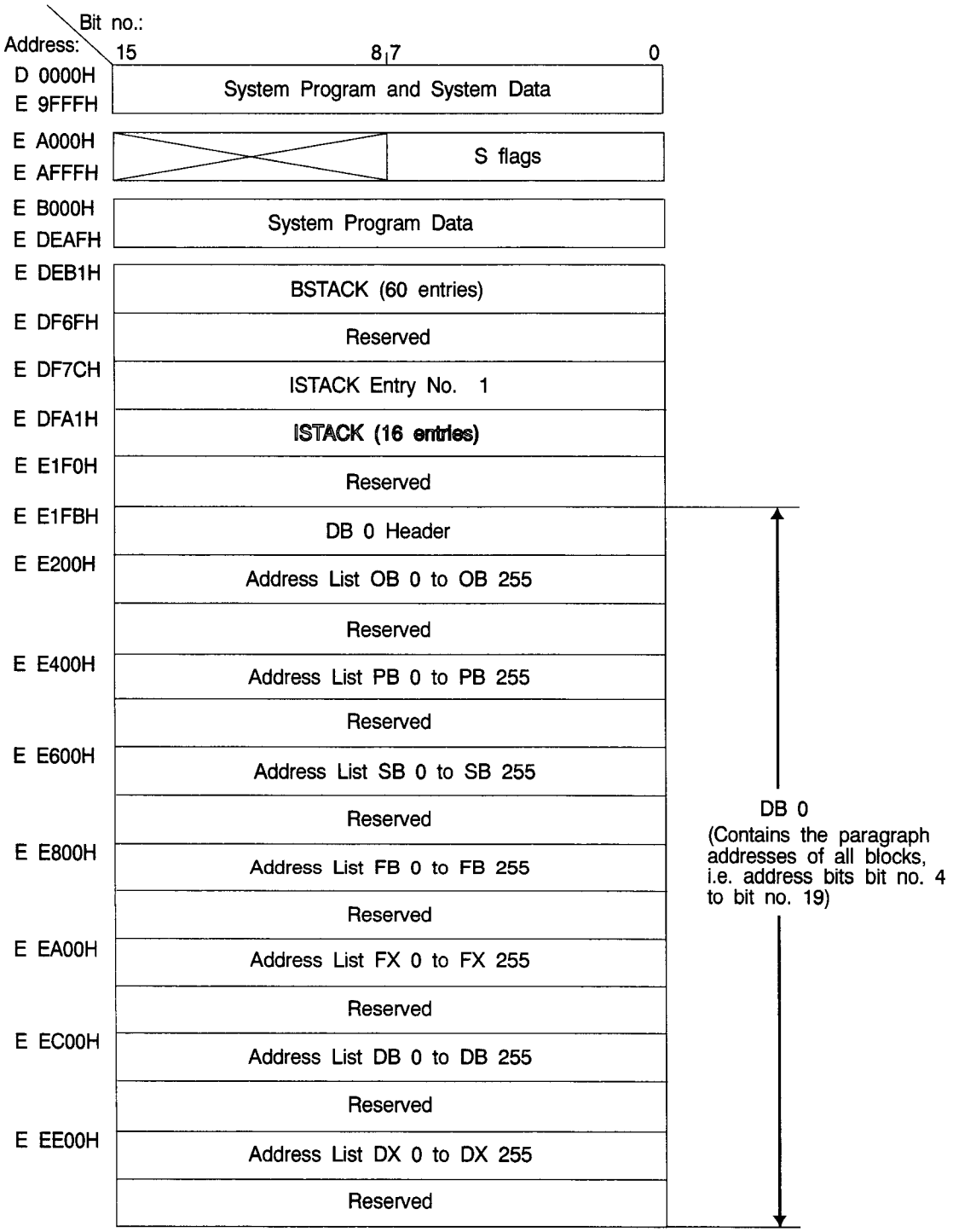


Fig. 8-2 Memory assignment for the system RAM/Part 1

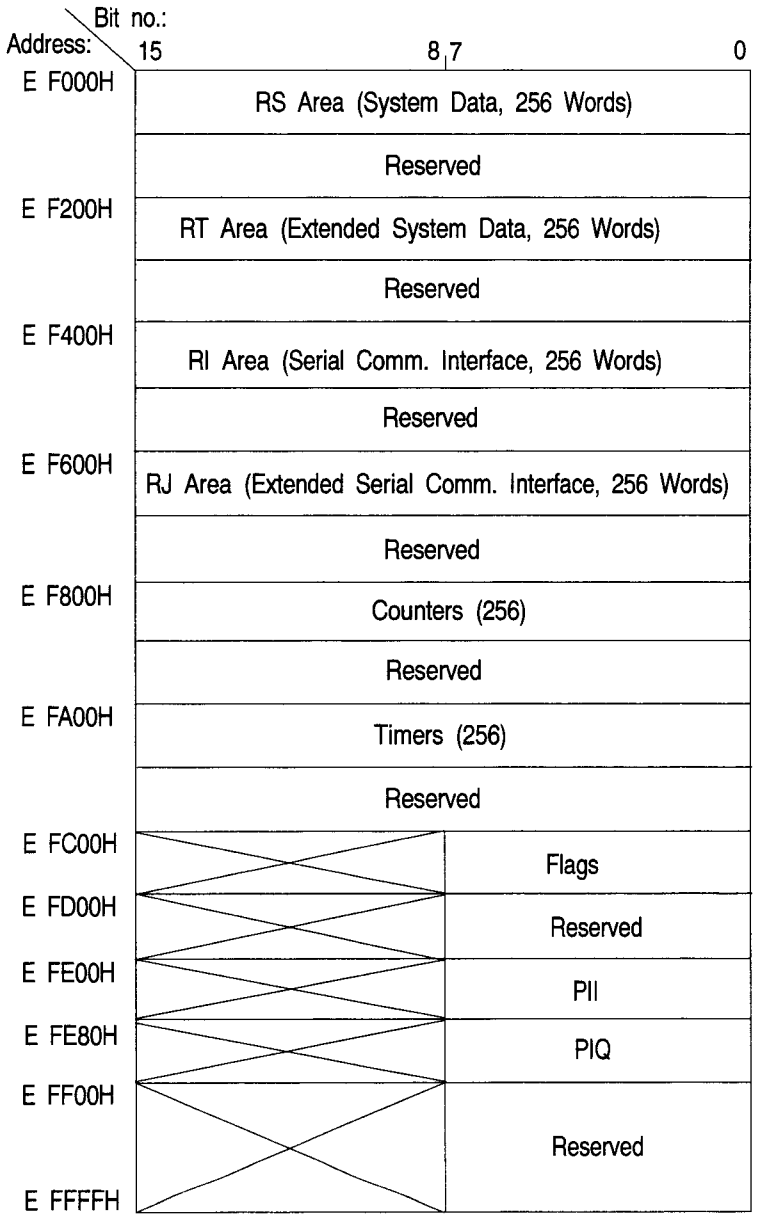


Fig. 8-3 Memory assignment of the system RAM/Part 2

8.2.2 Memory Assignment for the I/O

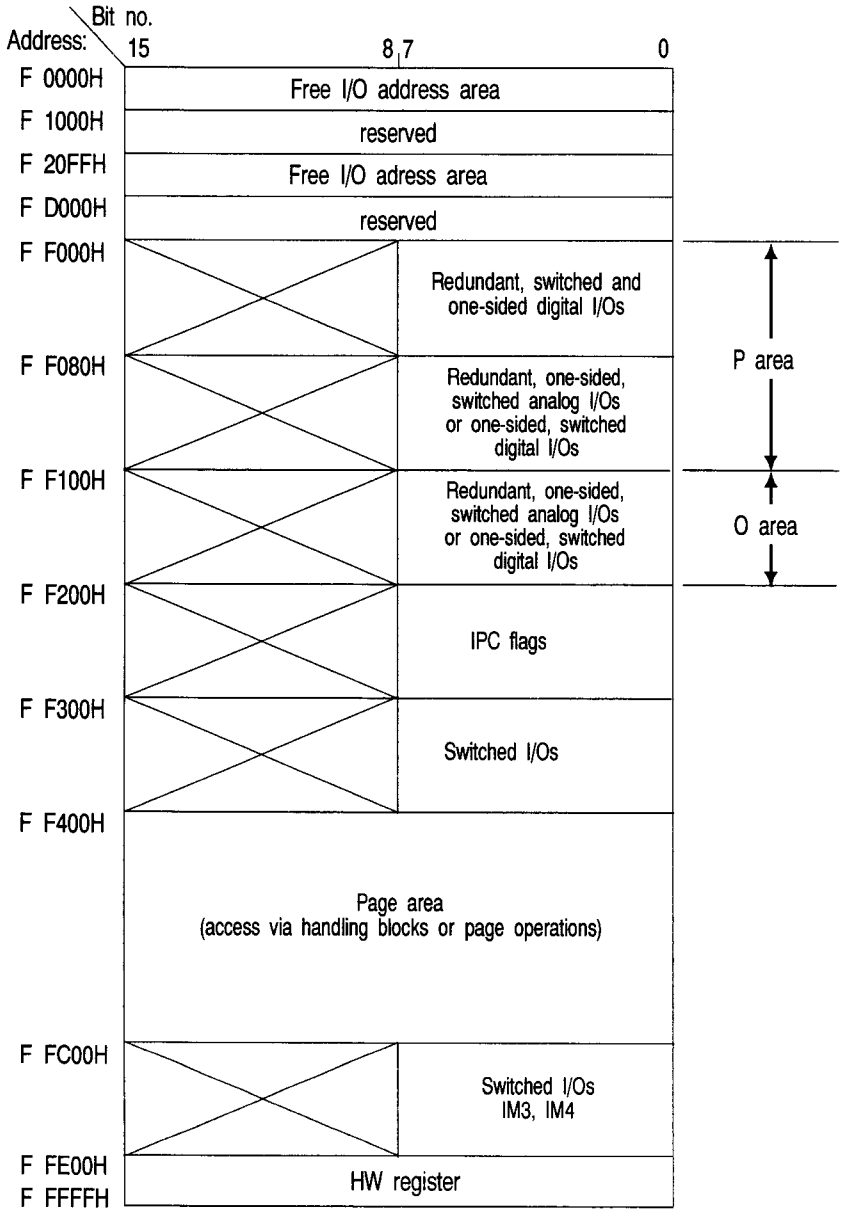


Fig. 8-4 Address areas for the I/O (8 bits) on the S5 bus

Address areas for the I/O and programming them

Area (absolute address)	Referenced with	Parameter
E FE00 PII (Process image input) E FE7F	L IB / T IB	0 to 127
	L IW / T IW	0 to 126
	L ID / T ID	0 to 124
	A I/ AN I / O I / ON I S I / R I / = I	0.0 to 127.7
E FE80 PIQ (Process image output) E FEFF	L QB / T QB	0 to 127
	L QW / T QW	0 to 126
	L QD / T QD	0 to 124
	A Q / AN Q / O Q / ON Q S Q / R Q / = Q	0.0 to 127.7
"P" peripherals with process image	When the operation is processed, only the process image is changed. The new status of the process image of the outputs is only output to the I/Os at the end of the cycle.	
F F000 Digital peripherals Inputs/ outputs F F07F	L PY / T PY	0 to 127
	L PW / T PW	0 to 126
F F080 Digital or analog peripherals Inputs/outputs F F0FF	L PY / T PY	128 to 255
	L PW / T PW	128 to 254
"P" peripherals	The inputs and outputs are addressed directly in bytes or words.	
F F100 Extended peripherals Inputs/outputs F F1FF	L OY / T OY	0 to 255
	L OW / T OW	0 to 254
"O" peripherals	The inputs and outputs are addressed directly in bytes or words.	

Using STEP 5 operations, you can access the I/O either directly or via the process image (PI). Note that a process image exists only for input and output bytes of the "P" peripherals with byte addresses from 0 to 127!

Note
Using the interface modules IM 304, IM 307 and IM 308, you can access IM3/IM4 address areas using your program. This allows access to two new address areas similar to the O area. In contrast to the O area, however, access to these areas is only possible using absolute addressing or using FB 196 of the "basic functions" software package (refer to Catalog ST59).

8

8.3 User Memory Organization in the CPU 948R

Depending on the version of the CPU 948R used, the user memory occupies the memory area from 0 0000H to C FFFFH. When you load the individual blocks of your program, they are stored in the memory in random order (with addresses in ascending order).

Block management

When you correct a block, the old block in the memory is declared invalid (i.e., the start ID is overwritten) and a new block is entered in the memory and the address list. This also applies when you delete blocks. The blocks are not really deleted in the memory but simply declared invalid. Gaps created by deleting are managed as available memory locations and are used again when you load new blocks.

Compressing memory

The online COMPRESS MEMORY PG function pushes all valid blocks in the memory together. When you activate the COMPRESS MEMORY while the CPU is in the **STOP** mode, all blocks that are not directly next to each other are shifted. However, when you activate this function while the CPU is in the **RUN** mode, long data and extended data blocks (i.e., longer than 512 data words) are not shifted because of data length consistency. Compressing produces large available memory areas which you can use for loading new blocks.

If the online COMPRESS MEMORY PG function is interrupted (e.g., when the power is turned off), compressing is terminated and does not resume automatically when power is turned on.

Location of blocks in the user memory

In the CPU 948R, blocks are stored so that data word DW 0 or the first STEP 5 statement of each block is located at a **paragraph address**. Paragraph addresses are at 16-word boundaries. Therefore, all blocks begin in the memory at the address xxxxBH (bit no. 0 to 3 = BH) and all block bodies at the address yyyy0H (bit no. 0 to 3 = 0H). The gaps that result between blocks are filled in by invalid data blocks, so that all blocks continue to exist in consecutive order.

Filler blocks

These invalid data blocks are known as "filler blocks". They are treated in the same way as the other blocks by the memory management and have the following structure:

Start ID:	7070H	;
Block type/block number:	01FBH	;DB 251 invalid
Programmer ID:	00FFH	;irrelevant
Library number:	FFFFH	;irrelevant
Block length:	00XXH	;length 5 to 20 words
Data:	FFFFH	;according
	:	;to
	:	;length;
	FFFFH	;can be left out entirely

Example

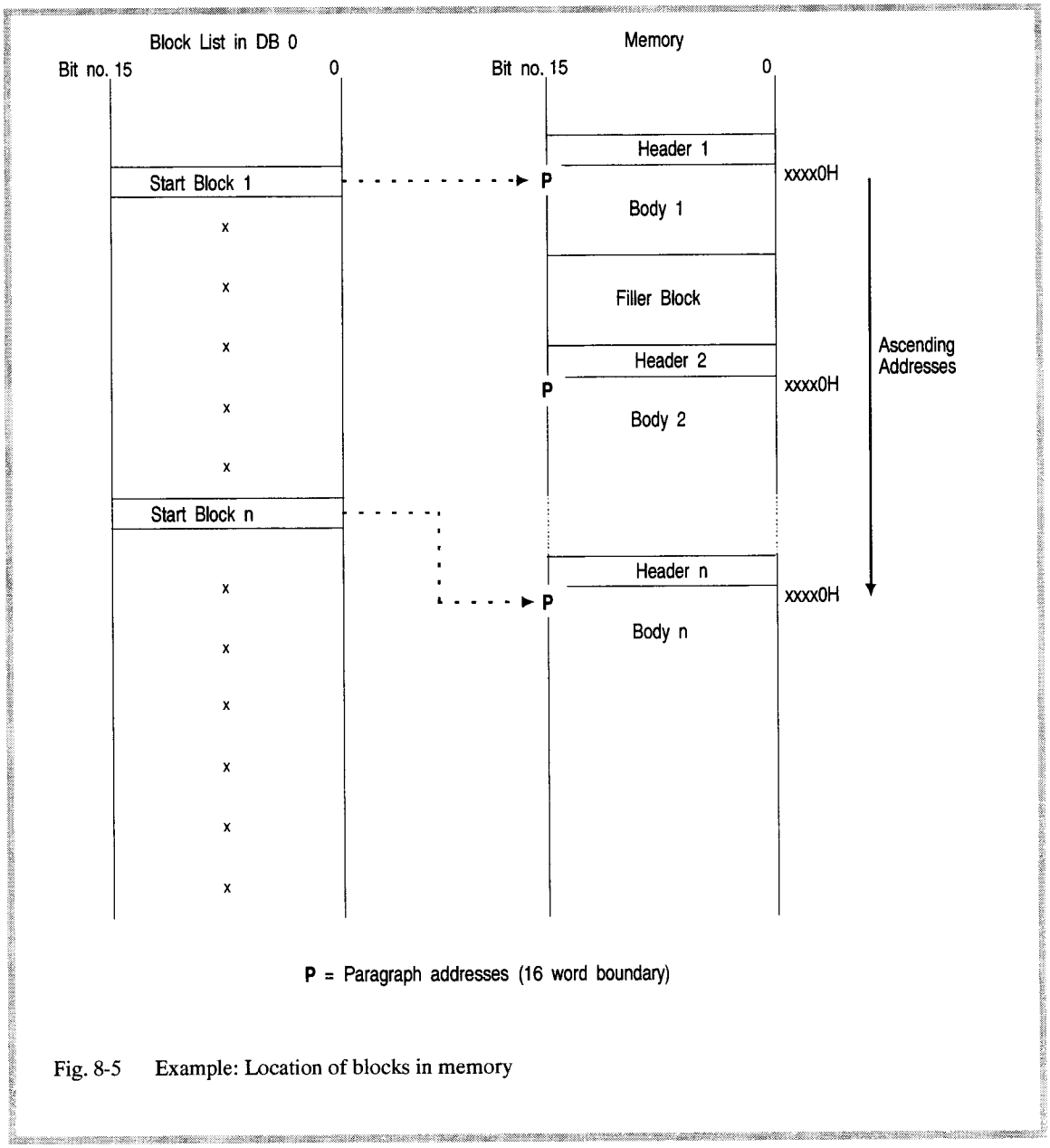


Fig. 8-5 Example: Location of blocks in memory

You can calculate the length of a filler block by finding the difference between the end address of the last block stored and the address before the next paragraph address.

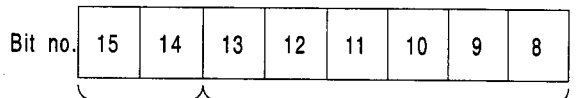
Difference	**Calculation for length of filler block (including header)**
0 to 5	Add 10 to the difference
6	No filler block is inserted
7 to 10	Add 10 to the difference
11 to 15	Subtract 6 from the difference

8.3.1 Block Headers in User Memory

Each block in the memory begins with a header that is five words long. The block header is divided as follows:

1st word: Block start ID: 7070H

2nd word: High byte = Block type



01H Data block DB
02H Sequence block SB
04H Program block PB
05H Function block FX
08H Function block FB
0CH Data block DX
10H Organization block OB

0 0 The block is invalid; it is not entered in the address list in DB 0.

0 1 The block in the RAM is valid; it is entered in the address list in DB 0.

Low byte = Block number
The low byte of the second header word contains the block number (0 to 255). It is coded as a hexadecimal number: 00 to FFH.

3rd word: The high byte of the third word contains the IDs for the programmer. The low byte contains part of the library number.

4th word: The fourth word contains the rest of the library number.

5th word: The fifth word (low and high bytes) contains the length of the block, including the block header. The length is indicated in words.

8.3.2 Block Address List in Data Block DB 0

Data block DB 0 is located in the system RAM of the CPU (beginning at address E E200H). It contains a list with the start addresses of all blocks in the user memory of the CPU. The system program generates (COLD RESTART) or checks (WARM RESTART) this list after power up; it updates it automatically when you use a programmer to enter or change blocks.

Address list start addresses

DB 0 has a separate, reserved address list of 256 words in each type of block. Blocks that are not loaded or have been deleted have the start address '0'.
The start addresses of each block address list are specified (see Section 8.2.1).

Block start addresses

The block start addresses in the address lists always point to the first word **after** the block header:

- with data blocks, to data word DW 0
- with logic blocks, to the first STEP 5 statement (in FBs to the 'JU' operation before the name and the parameter list).

Since each block is located at a paragraph address (16 word boundary), each address list entry in DB 0 is restricted to one word with bits number 4 to 20 of the address.

Location of block addresses in DB 0

n = E E400H (start address of the PB address list)

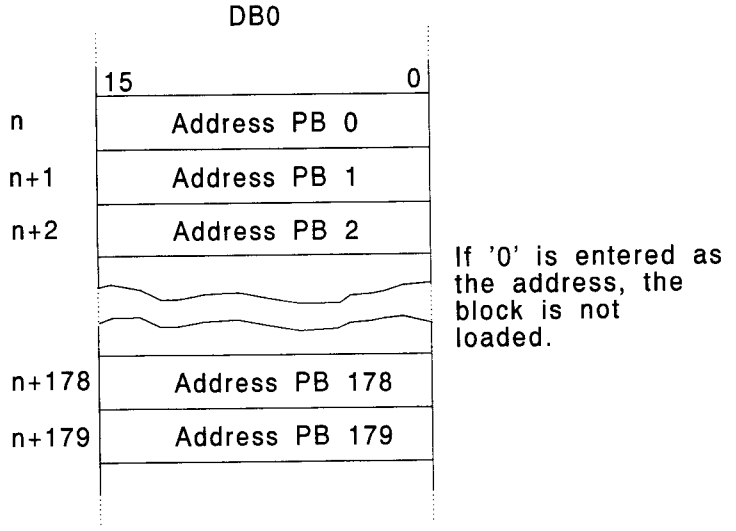


Fig. 8-6 Block addresses in DB 0

Example of how to obtain a block address

```
The block start addresses of the program blocks
are located in DB 0 and begin at address E E400H.
The start address of PB 22 can therefore be read
out by accessing memory at address E E416H (=
start address of the PB + 16H).
```

8.3.3 RI/RJ Area

The RI area is an area that is 256 words long in the internal system RAM of the CPU. RI occupies addresses E F400H to E F4FFH.

The RJ area is an area that is 256 words long in the internal system RAM of the CPU. RJ occupies addresses E F600H to E F6FFH.

You can use the entire RI area (RI 0 to RI 255) and the entire RJ area (RJ 0 to RJ 255) for your own purposes.

The RI/RJ area is only filled with zeros following OVERALL RESET.

8.3.4 RS/RT Area

The RS and RT areas contain information for the system programmer and system internal data.

The **RS area** is an area that is 256 words long in the internal system RAM of the CPU. RS occupies addresses E F000H to E F0FFH.



Caution
Only system data words RS 60 to RS 63 and bit RS 9.0 can be written.

The system data words reserved for the system program (on a dark shaded background in Table 8.2) **must not be read:**

System data words RS 50, RS 68 to 71 and RS 75 to 78 (on a light shaded background in Table 8.2) ***must only be read via the PG*, all other system data** (on a white background in Table 8.2) **can also be read by the user program.**

Illegal access to the system data area can affect the functional capability of your programmable controller and connected programmers: serious disturbances can occur which may put both people and machines in danger.

The **RT area** is an area that is 256 words long in the internal system RAM of the CPU. RT occupies addresses E F200H to E F2FFH.

You **cannot** use the **whole RT area** (RT 0 to RT 255) for your own purposes.

Only an overall reset can clear the RS/RT areas.

The information contained in some system data (about the internal structure of the CPU, the software release, the CPU ID etc.) is also available using the PG function SYSTEM PARAMETERS.

Assignment of the RS area

Table 8-2 Assignment of the RS area

RS	Name	Address
0	Input byte IB 0 image table (process interrupts)	E F000H
1	Process interrupts currently pending (IB 0)	E F001H
2 to 4	System program	
5	Current cycle time of the S5-155H	E F005H
6	System program	
7	STOP IDs (ISTACK)	E F007H
8	Start/restart IDs (ISTACK)	E F008H
9	QVZ test (bit 9.0)	E F009H
10 to 15	System program	
16	Error area: output bytes 0 to 15	E F010H
17 to 23	Error area: output bytes 16 to 127	E F011H to E F017H
24 to 31	Error area: input bytes 0 to 127	E F018H to E F01FH
32 to 47	Error area: interprocessor communication flag bytes 0 to 255	E F020H to E F02FH
48 to 59	System program	
60 to 63	Free for user	E F03CH to E F03FH
64 to 67	System program	
68 to 71	Error address for QVZ and PARE errors (must only be read via PG)	E F044H to E F047H
72 to 74	System program	
75 to 81	System message (must only be read via PG)	E F04BH to E F04EH

RS	Name	Address
Table 8-2 continued		
82 to 83	Interrupt code reset word (UALW)	E F052H to EF053H
84 to 250	System program	
251	Current cycle time of S5-155H	E F0FBH
252 to 255	System program	

Access by the STEP 5 user program to the system data words reserved for the system program causes RAM comparison errors or S5-155H synchronization errors. All these words can be read using the PG.

As a supplement to the table above, the following pages provide the bit assignments of some of the system data that you can evaluate using STEP 5 operations or with your programmer (see Section 5.4 for information on the abbreviations).

8.3.5 Bit Assignment of the System Data Words

System data RS 0

Input byte IB 0 Image table (external process interrupts)

Address E F000H

Table 8-3 Bits in RS 0 (image of IB 0)

High byte	
Bit no.	Assignment
15	Occupied by system program
14	
13	
12	
11	
10	
9	
8	
Low byte	
7	Status of I 0.7
6	Status of I 0.6
5	Status of I 0.5
4	Status of I 0.4
3	Status of I 0.3
2	Status of I 0.2
1	Status of I 0.1
0	Status of I 0.0

System data RS 1

Condition code of external process interrupts currently in processing queue

Address: E F001H

Table 8-4 Bits of RS 1 (current process interrupts)

High byte	
Bit no.	Assignment
15	All the bits have the value '0'
14	
13	
12	
11	
10	
9	
8	
Low byte	
7	Bit = '1': edge I 0.7
6	Bit = '1': edge I 0.6
5	Bit = '1': edge I 0.5
4	Bit = '1': edge I 0.4
3	Bit = '1': edge I 0.3
2	Bit = '1': edge I 0.2
1	Bit = '1': edge I 0.1
0	Bit = '1': edge I 0.0

System data RS 5

Current cycle time of the S5-155H

Address: E F005H

Table 8-5 Bits of RS 5 (current cycle time of the S5-155H)

High byte and low byte	
Bit no.	Assignment
15	The entered binary value * 10 ms equals the cycle time of the cycle processed last
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	

Example

Bit no.	15	14	13	12	11	10	9	8	7	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

The time of the last cycle is as follows:

$(2^4 + 2^3) * 10\ ms = (16 + 8) * 10\ ms = 240\ ms$

System data RS 7

Programmable controller STOP mode IDs (ISTACK)

Address: E F007H

Table 8-6 Bits of RS 7 (PLC stop IDs)

High byte	
Bit no.	Assignment
15	Reserved
14	
13	
12	Faulty ISTACK level
11	Illegal start-up type (UANL)
10	Interruption in stop loop
9	Illegal call of system block (SYSFHL)
8	Error in start-up block (AFEL)
Low byte	
7	Interruption by system (USYS –warm restart possible)
6	Interruption by programming error (UPROG – cold restart)
5	"End of program test" (BEARBE)
4	Stop switch (STOPS)
3	End of operation stop (STS)
2	End of cycle stop (STP)
1	reserved
0	PG stop (PGSTP)

8

System data RS 8

Start and restart IDs (ISTACK)

Address: E F008H

Table 8-7 Bits of RS 8 (start and start-up IDs)

High byte	
Bit no.	Assignment
15	Default: MANUAL COLD RESTART/ WARM RESTART (MSEG)
14	Default: AUTOMATIC COLD RESTART (ANEG)
13	Default: AUTOMATIC WARM RESTART (AWEG)
12	OVERALL RESET required (URLER)
11	WARM RESTART permitted (WIEZU)
10	COLD RESTART permitted (NEUZU)
9	OVERALL RESET executed (URLDF)
8	WARM RESTART executed (WIEDF)
Low byte	
7	COLD RESTART executed (NEUDF)
6	Automatic start after NAU
5	Manual start
4	COLD RESTART WITH MEMORY
3	PG overall reset
2	PG system start
1	PG warm restart
0	PG cold restart

System data RS 9

QVZ test

Address: E F009H

Table 8-8 Bits of RS 16 (error area output bytes 0 to 15)

High byte	
Bit no.	Assignment
15	Occupied by system program
14	
13	
12	
11	
10	
9	
8	
Low byte	
7	Occupied by system program
6	
5	
4	
3	
2	
1	
0	QVZ test

System data words RS 16 to RS 47

Error areas

RS xx	Address(es)	Error area
RS 16	E F010	Output bytes 0 to 15
RS 17 to RS 23	E F011 to E F017	Output bytes 16 to 127
RS 24 to RS 31	E F018 to E F01F	Input bytes 0 to 127
RS 32 to RS 47	E F020 to E F02F	Interprocessor communication flag bytes 0 to 255

RS 16

Address: E F010H

Table 8-9 Bits of RS 9 (QVZ test)

High byte	
Bit no.	Assignment
15	Output byte 0
14	Output byte 1
13	Output byte 2
12	Output byte 3
11	Output byte 4
10	Output byte 5
9	Output byte 6
8	Output byte 7
Low byte	
7	Output byte 8
6	Output byte 9
5	Output byte 10
4	Output byte 11
3	Output byte 12
2	Output byte 13
1	Output byte 14
0	Output byte 15

If errors appear during update of the process image input/output tables or interprocessor communication flags, the corresponding bits are set to '1'. – The system data words RS 17 to 47 are structured analogous to RS 16.

Example of RS 16

```
The content of system data register RS 16  is

   "8020" hexadecimal or "1000 0000 0010 0000"
   binary.


The process image for output bytes 0 and 10 has
not been updated correctly.
```

System data words RS 68 to RS 71

Error addresses of QVZ and PARE errors

If a QVZ or PARE error occurs, the address at which the error was detected is entered here.

	15 ... 0	
RS 68	QVZ error addr. high	E F044H
RS 69	QVZ error addr. low	E F045H
RS 70	PARE error addr. high	E F046H
RS 71	PARE error addr. low	E F047H

8

System data words RS 75 to RS 78

System message

The entries in system data words RS 75 to RS 78 refer to the error that occurred last. The message consists of four system data words with the following structure:

	15		0
RS 75	Error number	Parameter type	E F04BH
RS 76	Parameter 1		E F04CH
RS 77	Parameter 2		E F04DH
RS 78	Parameter 3		E F04EH

RS 75

The **high byte** contains the **error number** which classifies the error. The error number assigns the error to one of the following three areas:

Error groups

- 01H to 2FH: user error
- 30H to 3FH: error in DX 0 or DB 1,
- 40H system error

Parameter type

The **low byte** contains the **parameter type** that describes the structure of the succeeding parameter block in RS 76 to RS 78. The parameter types from 00H to 10H exist. The structure of the parameter field is described later.

The two following tables list the error groups "general errors" and "errors in DX 0 or DB 0".

General errors

Table 8-10 RS 75: general errors

Error number	Parameter type	Meaning
01H	01H	Block called is not loaded
02H	01H	Addressing error
03H	01H	Cycle time error
04H	01H	Substitution error
05H	02H	Timeout distributed peripherals
06H	03H	Timeout user memory
07H	01H	Load/transfer error with data blocks and extended data blocks
08H	01H	Bracket counter overflow
09H	04H	Opened data block does not exist
0AH	05H	Error with internal time interrupts/ other interrupts
0BH	03H	Timeout page frame area
0CH	03H	Timeout global communication area
0DH	07H	Timeout in process image updating
0EH	03H	Timeout in IPC flag synchronization error
0FH	03H	Timeout "P"/"O" peripherals
10H	03H	Timeout since interface module (IM 3/IM 4) missing
11H	03H	Parity error in user memory
12H	01H	Timeout process image transfer
13H	01H	Timeout input byte IB 0
14H	01H	BSTACK overflow
15H	01H	STS operation
16H	01H	RUN/STOP switch set to STOP position
17H	01H	Halt signal from the coordinator
18H	01H	Not used
19H	01H	Load/transfer error with L BY/T BY (process image update)
1AH	01H	Load/transfer error during addressing via the BR register
1BH	01H	I/Os not ready
1CH	08H	Timeout/parity error during initialization

Error number	Parameter type	Meaning
Table 8-10 continued:		
1DH	08H	AUTOMATIC WARM RESTART not possible; COLD RESTART required
1EH	00H	Illegal start-up type
1FH	01H	Load/transfer error during block transfer operation (incorrect memory area boundaries with TNW, TXB, TXW)
20H	09H	Illegal length for G DB/GX DX
21H	09H	DB/DX already exists for G DB/GX DX
22H	09H	Memory space insufficient for G DB/GX DX
23H	05H	Masked system interrupt is coming through
24H	00H	Block function (compress, transfer, input) required in STOP, COLD RESTART
25H	00H	Battery failure; no start-up possible
26H	00H	Change from single to multiprocessor operation prevents a WARM RESTART
27H	01H	STOP caused by STP operation
28H	01H	Continuous ready signal (I/O module defect))
29H	08H	Error in DB 0 struct. after OVERALL RESET

DX 0/DB 1 errors

Table 8-11 RS 75: Errors in DX 0 or DB 1

Error number	Parameter type	Meaning
Errors in DX 0		
30H	00H	No DX 0 in multiprocessing
32H	00H	Proc. int. and sys. int. selected simultaneously
33H	00H	Interrupts and mode incompatible
34H	06H	Invalid DX 0 header
35H	07H	Error in DX 0 block ID
36H	07H	Error in DX 0 parameter
Errors in DB 1		
38H	06H	No DB 1 in multiprocessing
39H	06H	Invalid DB 1 header
3AH	07H	DB 1 ID set more than once
3BH	07H	DB 1 byte offset without ID

Error number	Parameter type	Meaning
Table 8-11 continued		
3CH	07H	Peripheral entered in DB 1 is not plugged in
3DH	07H	Offset too big (parameter error)
3EH	07H	Too many offsets

Structure of the parameter field (RS 76 to RS 78)

Table 8-12 RS 76 to RS 78: Parameter types

Parameter type	Structure of the parameter field		
00H	No parameter; parameter 1, 2, 3 = 0		
01H	Parameter 1:	block type/block number (IDs from block header)	
	Parameter 2:	operation that caused an interruption	
02H	Parameter 1:	interface module number (IM 302) (distributed periphery)	
	Parameter 2:	number of the defective interface	
	Parameter 3:	incorrect byte offset	
03H	Parameter 1:	not used	
	Parameter 2:	error address high	
	Parameter 3:	error address low	
04H	Parameter 1:	block type/block number (IDs from block header)	
	Parameter 2:	operation that caused the interruption	
	Parameter 3:	number of the opened DB/DX	
05H	Parameter 1:	The following bits are set depending on the error:	
		Bit no. 0	= 1: timed interrupt/period 1
		Bit no. 1	= 1: timed interrupt/period 2
		: :	: :
		Bit no. 7	= 1: timed interrupt/period 8
		Bit no. 8	= 1: timed interrupt/period 9
		Bit no. 9	= 1: clock masked (ignored) for too long = 0: queue overflow
		Bit no. 10	reserved
		Bit no. 11	reserved
		Bit no. 12	= 1: interrupt G
		Bit no. 13	= 1: interrupt F
		Bit no. 14	= 1: interrupt E
		Bit no. 15	= 1: interrupt X

Parameter type	Structure of the parameter field		
Table 8-12 continued			
06H	Parameter 1	data word expected in DX 0 or DB 1 header	
	Parameter 2	actual data word in DX 0 or DB 1 header	
07H	Parameter 1	block ID or code word in DX 0 or DB 1 "DI08": input switched "DI01": input in subunit A "DQ08": output switched "DQ01": output in subunit A "CE00": IPC input flag "CA00": IPC output flag	
	Parameter 2	incorrect parameter 1 in DX 0 or DB 1 (FFFFH: parameter irrelevant)	
	Parameter 3	incorrect parameter 2 in DX 0 or DB 1 (FFFFH: parameter irrelevant)	
08H	Parameter 1	The following bits are set depending on the error:	
		Bit no. 0	= 1: QVZ in initialization
		Bit no. 1	= 1: PARE in initialization
		Bit no. 2	= 1: content of the memory card too large
		Bit no. 3	= 1: operating system error
		Bit no. 4	= 1: incorrect block ID
		Bit no. 5	= 1: incorrect block delimiter
		Bit no. 6	reserved
		Bit no. 7	reserved
		Bit no. 8	= 1: DB 0 changed since last COLD RESTART
		Bits no. 9 to 15	reserved
	Parameter 2	error address high, (when bit 2, 4 or 5 of parameter 1 = '1'	
	Parameter 3	error address low, (when bit 2, 4 or 5 of parameter 1 = '1'	
09H	Parameter 1	opcode "GX DX" or "G DB" (indicates the block type)	
	Parameter 2	block number	
	Parameter 3	data block length	
10H	internal system error number		
0AH	Parameter 1	block type/block number (IDs from block header)	
	Parameter 2	expected checksum	
	Parameter 3	actual checksum	
0BH	Parameter 1	FFFFH	
	Parameter 2	error address high with address code error, actual checksum high when testing the system program code	
	Parameter 3	error address low with address code error, actual checksum low when testing the system program code	

Parameter type	Structure of the parameter field	
Table 8-12 continued		
0CH	Parameter 1	pattern for testing the user memory
	Parameter 2	error address high
	Parameter 3	error address low

Example of a system message

RS 75	21H	09H	E F04BH
RS 76	7804H		E F04CH
RS 77	0064H		E F04DH
RS 78	0078H		E F04EH

```
RS 75, error number = 21H: The error occurred in the STEP 5 user program when
                           generating a DB/DX.
RS 75, parameter type=09H: The parameter block in RS 76 to RS 78 contains
                           3 parameters.
RS 76, parameter 1 = 7804H: Opcode = "GX DX", this means block type "DX".
RS 77, parameter 2 = 0064H: Block number = 100 (dec.)
RS 78, parameter 3 = 0078H: Data block length = 120 data words

Information contained in the message:
In the STEP 5 user program, data block DX 100 should be generated with a
length of 120 data words. However, this already exists.
```

System data RS 251

Current cycle time of the S5-155H

Address: E F0FBH

Table 8-13 Bits of BS 251 (current cycle time of the S5-155H)

High byte and low byte	
Bit no	Assignment
15	The entered binary value * 10 ms = the cycle time of the cycle processed last
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	

8.3.6 Addressable System Data Area

The system program uses the memory area from E 8200H to E DEF0H as an addressable system data area.

PLC identification field

At the start of this area there is an information field of 12 words in which an identifier of the PLC is entered.

This field has the following structure:

Word			
0	'S'	'5'	E 8200H
1	'1'	'5'	E 8201H
2	'5'	'U'	E 8202H
3	'C'	'P'	E 8203H
4	'U'	'9'	E 8204H
5	'4'	'8'	E 8205H
6	'R'	'x'	E 8206H
7	'.'	'y'	E 8207H
8	0		
9	0		
10	0		
11	0		E 820BH

For 'x' and 'y' the current version number is entered.

System parameters

System parameters in the memory area beginning with the address E 8210H:

Word			
0	Start add. interface module input		E 8210H
1	Start add. interface module output		E 8211H
2	Start add. process image input table		E 8212H
3	Start add. process image output table		E 8213H
4	Start add. flags		E 8214H
5	Start add. timers		E 8215H
6	Start add. counters		E 8216H
7	Start add. system data		E 8217H
8	Status ID	PLC software version	E 8218H
9	User memory end address		E 8219H
10	System program memory		E 821AH
11	Length of DB list		E 821BH
12	Length of SB list		E 821CH
13	Length of PB list		E 821DH
14	Length of FB list		E 821EH
15	Length of OB list		E 821FH
16	Length of FX list		E 8220H
17	Length of DX list		E 8221H
18	Length of DB address list (DB 0)		E 8222H
19	Slot ID (see below)	CPU ID 2 (see below)	E 8223H
20	Block header length		E 8224H
21	CPU ID 1 (see below)	Programmer interface software version	E 8225H

You can also use the SYSTEM PARAMETER PG online function to find the information contained in a few system data registers (e.g., concerning the internal structure of the CPU, the software version, the CPU ID).

Word 19 and word 21

Structure of words 19 and 21:

Word 19

Bit no.	High byte
15	0
14	0
13	0
12	0
11	0
10	0
9	0
8	Slot ID CPU 1
	Low byte
7	CPU type: 0011 = S5-155H/CPU 948R (only valid in conjunction with the CPU ID)
6	
5	
4	
3	CPU-ID 2: 1000 = S5-155U
2	
1	
0	

Word 21

Bit no.	High byte
15	reserved
14	
13	
12	
11	
10	
9	Bit 8 = 0, bit 9 = 1 for "150U"
8	
Bit no.	**Low byte**
7	Release of the PG interface software in the form "xyH" Example: 13H corresponds to release "V1.3"
6	
5	
4	
3	
2	
1	
0	

Memory Access Using Absolute Addresses

9

Contents of Chapter 9

9.1	Introduction	9 - 4
9.2	Memory Access via Addresses in ACCU 1	9 - 8
9.2.1	LIR/TIR: Loading to or Transferring from a 16-Bit Memory Area Indirectly	9 - 9
9.2.2	Examples of Access to DW > 255	9 - 15
9.2.3	LDI/TDI: Loading to or Transferring from a 32-Bit Memory Area Indirectly	9 - 17
9.3	Transferring Memory Blocks	9 - 19
9.4	Operations with the Base Address Register (BR Register)	9 - 22
9.4.1	Operations for Transfer between Registers	9 - 23
9.4.2	Accessing the Local Memory	9 - 24
9.4.3	Accessing the Global Memory	9 - 25
9.4.4	Accessing the Page Memory	9 - 29

9

Memory Access Using Absolute Addresses

This chapter explains how to use STEP 5 operations and special STEP 5 registers to address data in certain memory areas using absolute addresses.

9.1 Introduction

The STEP 5 programming language contains operations with which you can access the entire memory area. These operations belong to the "system operations".

The operations described in this chapter work with 20-bit absolute addresses. Consequently, they are dependent on the memory size and type, the peripherals, CPs, and IPs of your programmable controller.



Warning
If the operations described in this chapter are not used properly, STEP 5 blocks and system data can be overwritten. Therefore, only experienced programmers should use operations that work with absolute addresses.

Local memory

Local memory is the memory area that is available in each CPU. It includes the following: user submodule, RI/RJ area, RS/RT area, counters, timers, flags, process images.

Global memory

Global memory exists only once for all CPUs. You address it via the S5 bus.

Memory organization

Memory areas are organized **in bytes** or **in words** as follows:

- Bytes: Each address addresses a byte.
- Words: Each address addresses a 16-bit word (= 2 bytes).

Organization of the **local** memory is fixed (see Chapter 8)

Organization of the **global** memory depends on the type of modules that are plugged into the programmable controller:

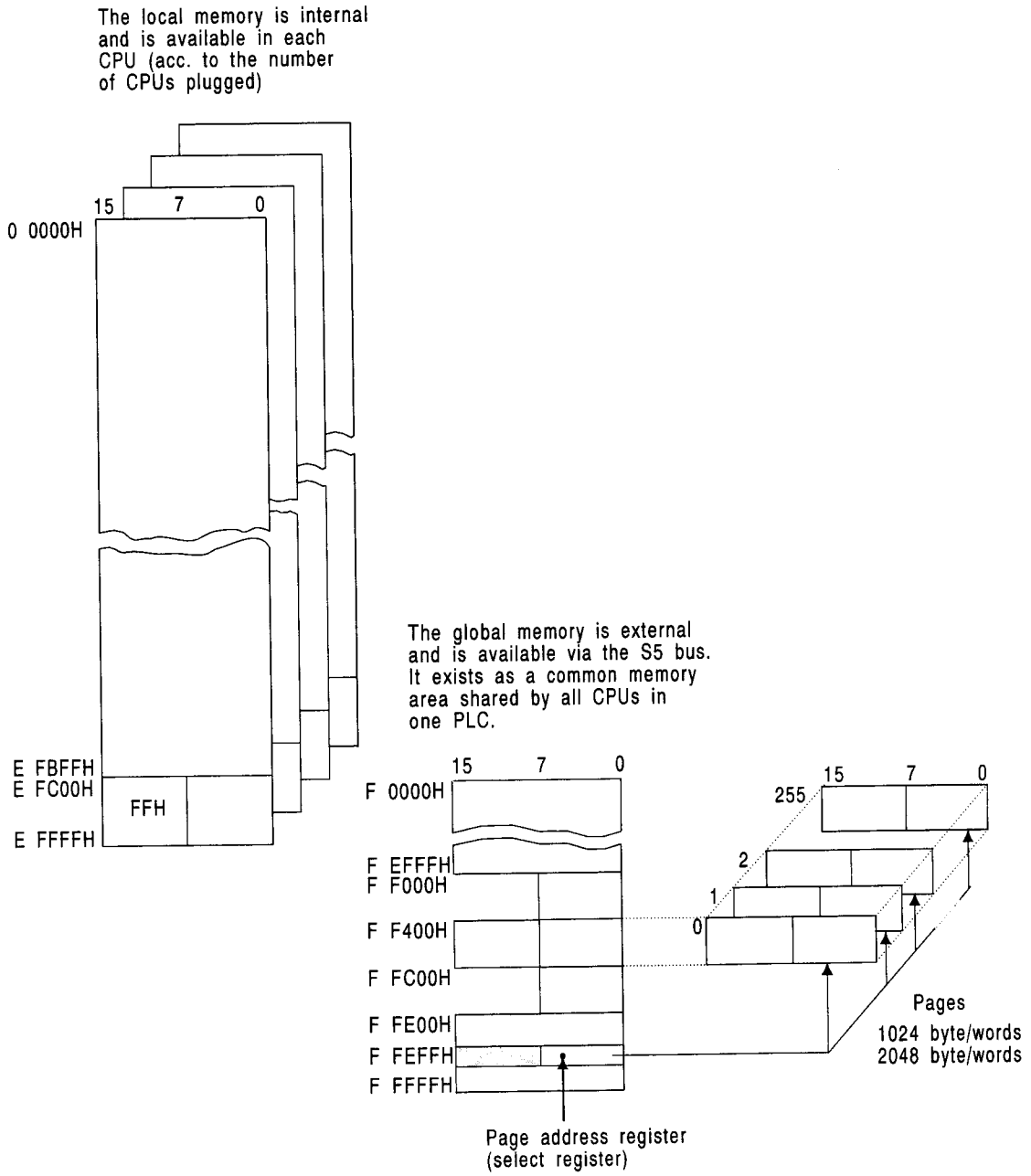


Fig. 9-1 Global and local memory

Memory access

Using absolute addresses, you can access the following local or global memory areas with the operations indicated (refer to Fig. 9-2).

Access to the local and global area

You can access both the local and global areas:

- Local area (addresses 0 0000H to E FFFFH) and global area (addresses F 0000H to F FFFFH) with:

  LIR, TIR, LDI, TDI, TNW, TXB, TXW,

- Section of the local area organized in words (addresses 0 0000H to E FBFFH) or in bytes (addresses E FC000 to E FFFF) with:

  LRW, TRW, LRD, TRD.

Access only to the global area

You can access the following parts of the global area:

- Section of the global area organized in bytes (addresses F 0000H to F FFFFH) with:

  LY GB, LY GW, LY GD, TY GB, TY GW, TY GD, TSG,

- Section of the global area organized in words (addresses F 0000H to F FFFFH) with:

  LW GW, LW GD, TW GW, TW GD, TSG .

Access to the page area

You can access the following parts of the page area:

- Section of the global area organized in bytes (addresses F F400H to F FBFFH, = dual-port RAM area) with:

  LY CB, LY CW, LY CD, TY CB, TY CW, TY CD, TSC,

- Section of the global area organized in words (addresses F F400H to F FBFFH, = dual-port RAM area) with:

  LW CW, LW CD, TW CW, TW CD, TSC

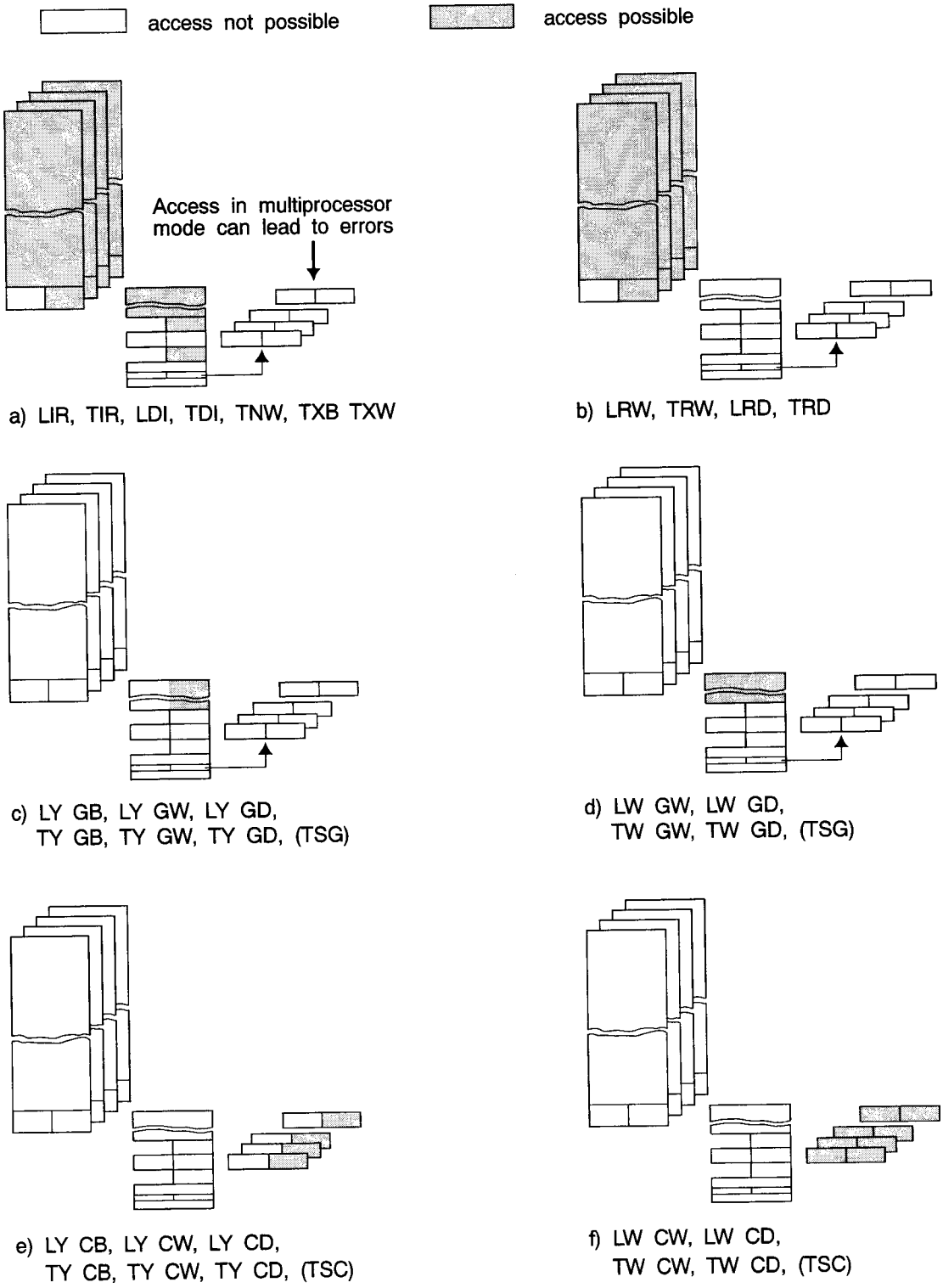


Fig. 9-2 Access to local or global areas using absolute addresses

9

9.2 Memory Access via Address in ACCU 1

Application

The operations listed in this section are suitable primarily for access to data blocks and other operand areas. You should, however, **not** access blocks containing STEP 5 programs (OBs, FBs, PBs and SBs) with these operations.

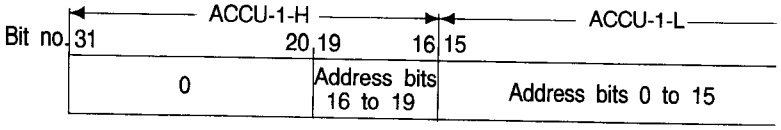
To access these areas, you can use various 16 or 32-bit wide registers. These registers include the accumulators ACCU 1 to ACCU 4 and other special registers used as resources by the CPU.

Operations

Table 9-1 Operations for indirect memory access using registers

Operation	Operand	Function
LIR	Register no. 0 to 15	Load the **16-bit register** with the content of a memory word addressed by ACCU 1 (20-bit address).
TIR	Register no. 0 to 15	Load the content of the **16-bit register** in the memory word addressed by ACCU 1 (20-bit address).
LDI	Register name	Load the **32-bit register** with the contents of the memory words 'n' and 'n+1' addressed by ACCU 1 (20-bit address).
TDI	Register name	Load the contents of the **32-bit register** in the memory words 'n' and 'n+1' addressed by ACCU 1 (20-bit address).

The absolute address of the memory word or the first of the two memory words is located in ACCU 1 in the following representation:



The following pages explain **which registers** you can use with the operations.

Examples explain **how** to use the operations.

9.2.1 LIR/TIR: Loading to or Transferring from a 16-Bit Memory Area Indirectly

The following table shows which register numbers you can use with the CPU 948R for the LIR and TIR operations and how these are assigned.

Table 9-2 16-bit register for LIR/TIR

Register no.	Register assignment (each 16 bits wide)
0	ACCU-1-H (left word of ACCU1, bits 16 to 31)[1)]
1	ACCU-1-L (right word of ACCU1, bits 0 to 15)[1)]
2	ACCU-2-H
3	ACCU-2-L
5	Block stack pointer (offset)
6	DBA (data block start address register)
8	DBL (data block length register)
9	ACCU-3-H
10	ACCU-3-L
11	ACCU-4-H
12	ACCU-4-L

1) Loading the contents of an addressed memory register into register '0'or '1 overwrites the address stored in ACCU 1.

Registers 4, 7, 13, 14 and 15 do not exist on the CPU 948R. LIR/TIR operations with these register numbers must not be used.

9

LIR/TIR: with 8-bit memory areas

If you use the LIR and TIR operations to access memory areas that are only 8 bits wide, remember that

- the LIR operation overwrites the high byte of the register with **non-defined values** (except flags, the PII and PIQ; in these areas, FFH is written to the high byte) .

  and

- the TIR operation transfers only the low byte of the register (the high byte of the register is lost).

Figs. 9-3 and 9-4 illustrate the difference when accessing word and byte-oriented memory areas using LIR/TIR.

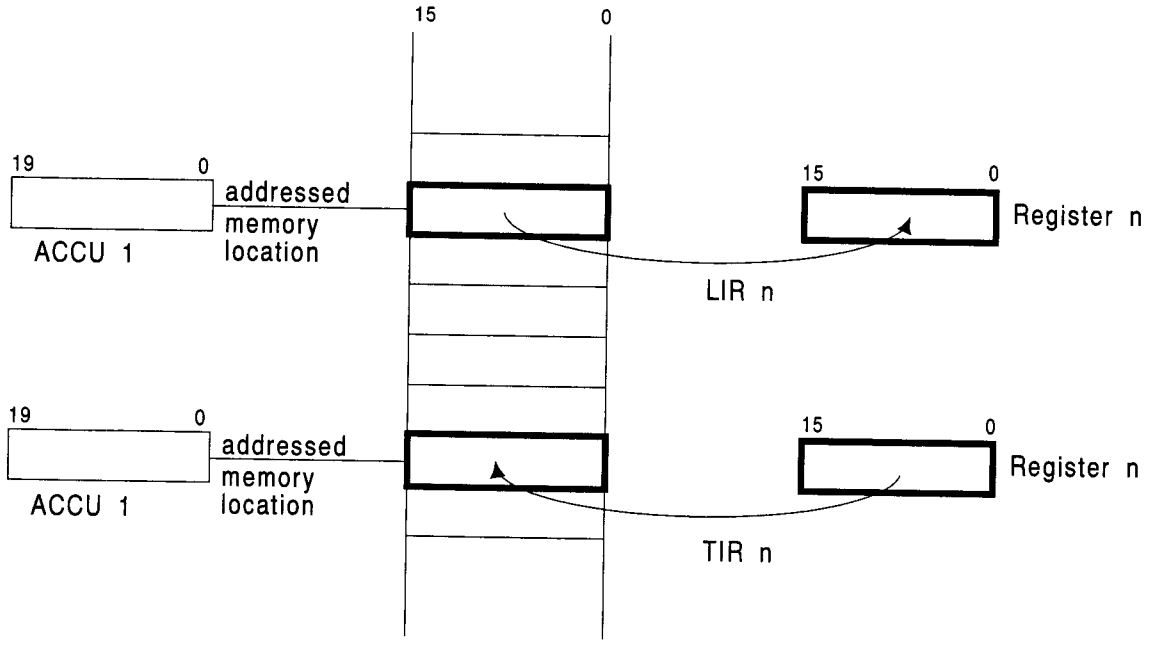


Fig. 9-3 LIR/TIR with 16-bit memory areas (word-oriented)

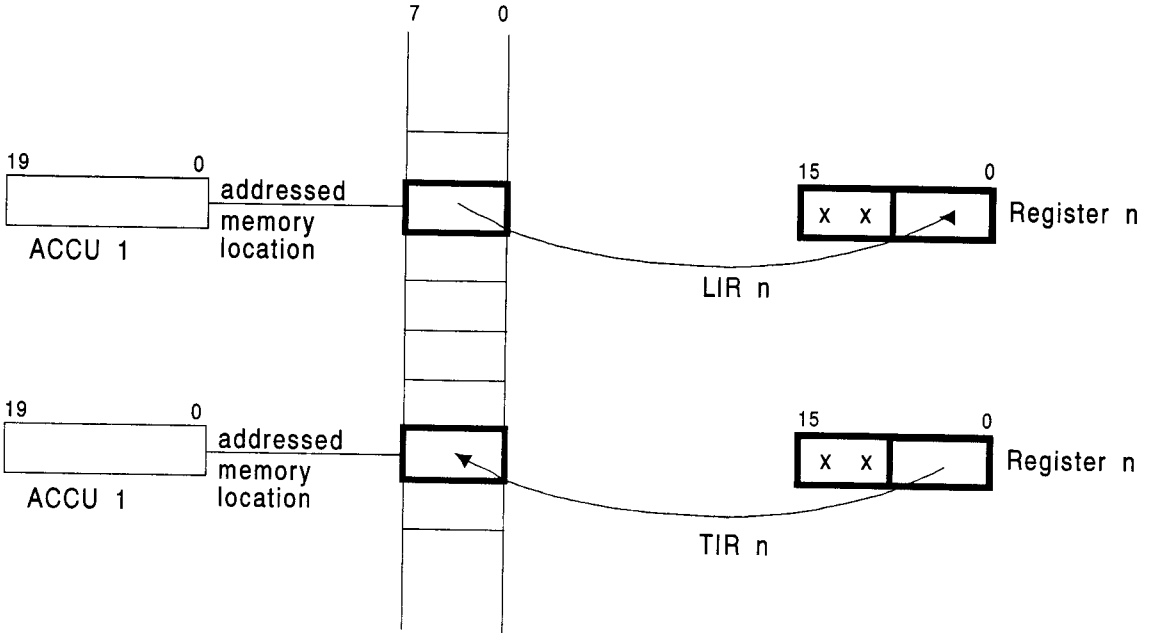


Fig. 9-4 LIR/TIR with 8-bit memory areas (byte-oriented)

Registers 0 to 3 and 9 to 12: ACCUs 1, 2, 3 and 4

During program processing, the accumulators are used as buffers for the CPU. The TIR operation transfers the contents of the accumulators into absolutely addressed memory registers. The LIR operation loads the contents of absolutely addressed memory registers into the accumulators. The absolute address of the memory locations is in ACCU 1, bit number 0 to 19.

Examples

```
The contents of the memory location with address E F800 are loaded in flag
word FW 100.

   :L    DH 000E F800 Load address E F800 of the memory location in ACCU 1
   :LIR  1            Load the contents of the memory location addressed by
   :                  ACCU 1 in register 1 (= ACCU-1-L)
   :T    FW 100       Store the contents of address E F800 in flag word FW 100
   :BE
```

```
The content of the flag word 200 is transferred to the memory location with
address E F800.

   :L    FW 200       Load flag word FW 200 in ACCU 1
   :L    DH 000E F800 Load address E F800 to which the data will be
   :                  transferred in ACCU 1 (flag word FW 200 to ACCU 2)
   :TIR  3            Transfer the contents of register 3 = ACCU-2-L in to the
   :                  memory location addressed by ACCU 1
   :BE
```

Register 6: DBA (Data Block Start Address)

When you open a data block or an extended data block using the C DB or CX DX operations, the address of DW 0 in the opened data block is loaded into register 6. The block address list in DB 0 contains this address.
The DBA register is set to '0' before each OB 1 call.

The DBA register **remains the same** if one of the following occurs:

- a jump operation (JU/JC) causes program processing to continue in a different block,

  or

- the CPU activates a different program processing level.

The DBA register **changes** if one of the following occurs:

- Another data block is opened,

  or

- the program returns to a higher order block after a new data block is opened in the called block (refer to Section 2.4.3).

Example

Effect of the "**CX DX 17**" operation on the DBA register:

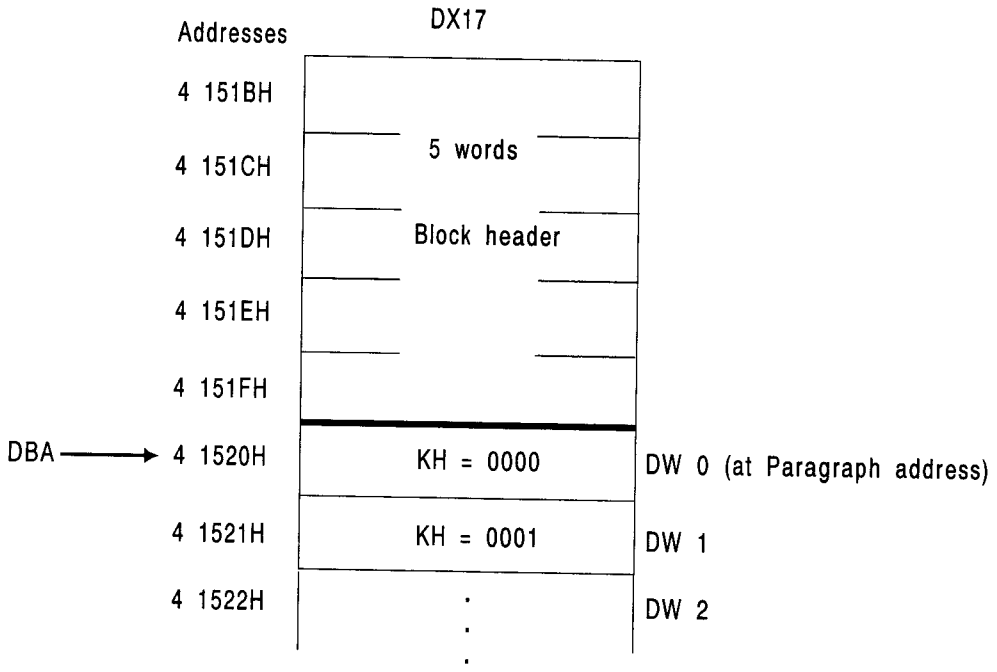


Fig. 9-5 Using the DBA register

When DX 17 is called, the address of the memory word in which DW 0 is stored is entered in the DBA register. In this example, the DBA is **4152H.**

Note: In the ISTACK, the address entered in the DBA register appears under the heading 'DB-ADD'.

Register 8:
DBL = Data Block Length

In addition to the DBA register, a DBL register is loaded every time a data block is called. It contains the length (in words) of the data block called, **without** the block header. The DBL register is set to '0' before each OB 1 call.

DBL register is **retained**, when

- program execution is continued in a different block following a jump statement (JU/JC)

  or

- a different program execution level is nested in.

It **changes** when

- a different data block is opened

  or

- program execution returns to a higher order block after a new data block is opened in the called block (refer to Section 2.4.2).

Note
You can change the DBA and DBL registers using LIR operations to address data block addresses higher than 255. The DBA register contains **paragraph addresses**. Make sure that a change in the DBA register does not automatically cause a change in the DBL register and vice-versa. **This would mean that transfer error monitoring is no longer guaranteed.**

On the CPU 948R, changes in the DBA/DBL registers are **undone**, as soon as the **current block is completed** (BSTACK entries). Manipulations on the DBA/DBL registers are therefore only effective in the block in which they were made.

Example

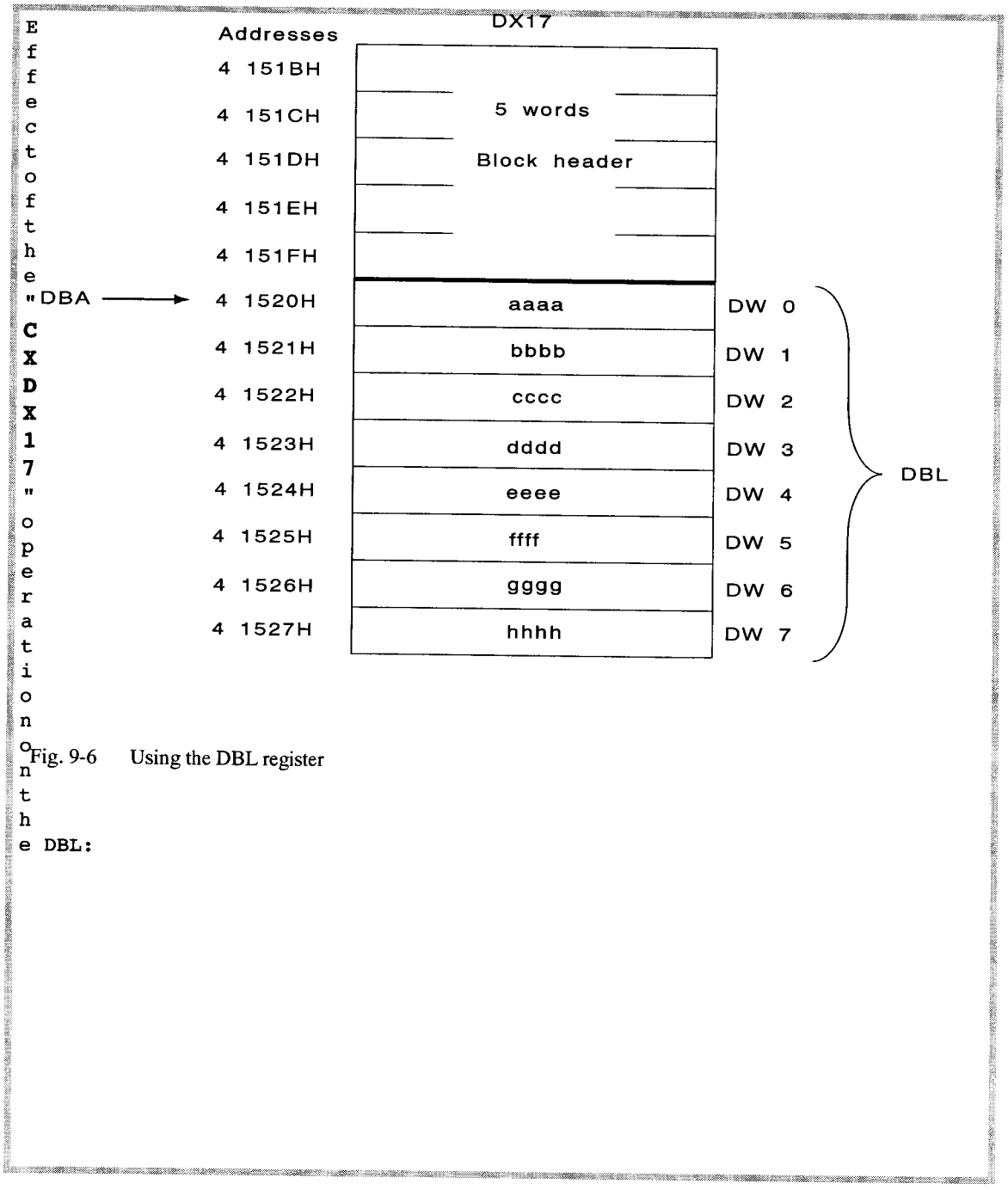


Fig. 9-6 Using the DBL register

Effect of the operation on the DBL:

"CXDX17"

9.2.2 Examples of Access to DW > 255

Example 1:

The content of data word DW 300 in DB 100 is read and transferred to flag word FW 100 (by changing the STEP 5 operation shown in bold face, it can also be used to read other data blocks (DB or DX)).

```
FB  5

SEGMENT 1          0000          Reading out DW 300 from DB 100
NAME :LIR DW

0005      :L   DH   000E EC00     Start addr. of the DB list
0008      :L   KF   +100           plus the DB number
000A      :+D                      = Address list entry of DB 100
000B      :                        (Bits 4 to 19)
000C      :LIR       1           Start addr. DB 100 to ACCU 1
000D      :SLD       4           Convert addr. to physical addr.
000E      :L   KF    +300        DW 300 is read out
0010      :+D                    Addr. = start addr. of DB + DW addr.
0011      :LIR       1           Store content of DW 300 to ACCU 1
0012      :T   FW 100              in FW 100
0013      :BE
```

Example 2:

All the data words of a data block will have a constant written to them.

The program shown below writes the constant KH = A5A5 to all data words of DB 100. After changing the STEP 5 operations shown in bold face, it can also be used to write values to other data blocks (DB or DX). Non-existent data blocks are detected and cause a jump to the NIVO marker.

The program uses three accumulators. Within the loop, the accumulator contents do not change.
ACCU 1 initially contains the address of the first data word and is incremented by one each time the loop is run through.
ACCU 2 contains the address of the last data word + 1. The loop is terminated as soon as the content of ACCU 1 is the same as the content of ACCU 2.

To write the data words, the operation TIR 10 is used which stores the content of ACCU-3-L (the constant) at the address contained in ACCU 1.

ACCU contents within the loop:

ACCU 1: address of the **current** data word to be written to
ACCU 2: address of the **last** data word to be written to **+ 1**
ACCU 3: constant

Continued on the next page

Example 2 continued:

Flag assignment:

```
FW 10:     Bits 4 to 19 of the start address of the DB/DX (points to DW 0)
FW 12:     Length of the DB/DX (number of data words)
FD 14:     Address of the last data word in the DB/DX + 1 (physical address)

FB  6

SEGMENT 1            0000          Writing a DB with a constant
NAME :FILL DB
0005        :                      ! Required flags: FY 10 to FY 17      !
0006        :L   DH   000E EC00    Start addr. of the DB list
0009        :L   KF   +100          plus the DB number
000B        :+D                     = Address list entry of DB 100
000C        :                       (Bits 4 to 19)
000D        :LIR        1          Start addr. DB 100 to ACCU 1
000E        :T   FW   10           Buffer start addr.
000F        :L   KB    0            (Paragraph address)
0010        :!=F                   If start addr. = 0, then
0011        :JC  =NIVO              DB does not exist
0012        :
0013        :L   FW   10           Start addr. of DB (1st DW)
0014        :SLD       4           Convert addr. to physical address
0015        :L   KB    1           Find out DB length via
0016        :-D                     5th word in block header
0017        :LIR       1           Length including block header to ACCU 1
0018        :ADD BN   -5           Number of DWs = total length - 5 words (header)
0019        :
001A        :T   FW   12           Buffer length
001B        :
001C        :L   FW   12           Number of data words +
001D        :L   FW   10            start address (DW 0 converted to
001E        :SLD       4            physical address)
001F        :+D                     produces
0020        :T   FD   14            address of the last DW + 1
0021        :
0022        :L   KH   A5A5         Constant, written to all data words
0024        :
0025        :L   FD   14           Address of the last DW + 1
0026        :ENT                   Shift constant to ACCU-3-L
0027        :                       (= register 10)
0028        :L   FW   10           Convert address of 1st data word (DW 0)
0029        :SLD       4            to physical address
002A        :
002B SCHL   :                      Loop:
002C        :                       ACCU 1: address of DW to be written to
002D        :                       ACCU 2: address of last DW + 1
002E        :                       ACCU 3: constant
002F        :TIR       10          Store the value of ACCU-3-L in the DW with
0030        :                       the address in ACCU 1
0031        :
0032        :ADD DH    0000 0001   Increment address by 1
0035        :
0036        :><D                   Scan whether last DW reached
0037        :JC  =SCHL              (if not, return to the loop)
0038        :
0039 WEIT   :                      Continue the program ...
003A        :                       after all DWs have been written to ...
003B        :
003C        :BEU
003D        :
003E NIVO   :                      If DB 100 does not exist
003F        :BE
```

9.2.3 LDI/TDI: Loading to or Transferring from a 32-Bit Memory Area Indirectly

The following table shows which register names you can use on the CPU 948R for the LDI and TDI operations and how these are assigned.

Table 9-3 32-bit register for LDI/TDI

Register name	Register assignment (each 32 bits wide)
A1	ACCU 1 (ACCU1, bits 0 to 31) [1]
A2	ACCU 2 (ACCU1, bits 0 to 31)
SA	STEP address counter (bits 0 to 19)
BA	BA register (start address of actual operands, bits 0 to 19)
BR	BR register (base address register, bits 0 to 19)

[1] Loading the contents of an addressed memory register into the A1 register overwrites the address stored in ACCU 1.

Byte addresses

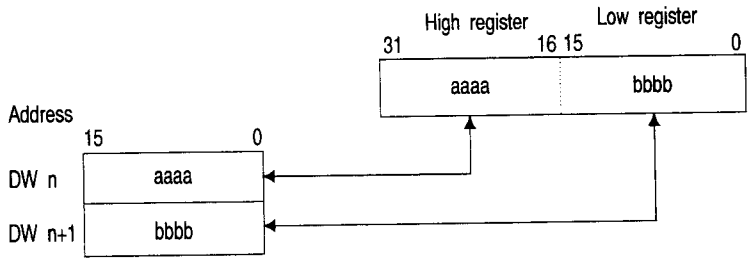
If you reference byte addresses with the LDI or TDI operations, note the following:

- With LDI, **non-defined values** are written to the high bytes of the referenced register (except flags, the PII and PIQ; in these areas, FFH is written to the high byte) ,

  and

- with TDI, only the low bytes of the referenced register are transferred (the high bytes are lost – refer to the example on the following page).

Data storage with LDI/TDI



SA Register:
SAC = STEP Address Counter

On completion of the operation, the 20-bit absolute address **of the operation** to be processed next is entered in the SA register.

BA Register:
Start Address of the Actual Operands

During program processing of a function block int the STEP 5 user program, a 20-bit absolute address of the actual operands in the higher order block is entered in the BA register.

BR Register:
Available Base Address Register

The base address register (20 bits) allows you to calculate addresses and to execute indirect load and transfer operations without using the ACCUs for the address. It can be used freely during STEP 5 program processing.

Example of TDI in the byte area

```
:L    DH 1234 5678    Load data
:L    DH 000E FC00    Load address of flag
:                     byte FY 0
:TDI  A2              Store content of ACCU 2

E FC00 = 34   (The values '12H' and '56H' from
E FC01 = 78   ACCU 2 are lost)
```

9.3 Transferring Memory Blocks

Application

With the operations explained in this section, you can re-store data areas with a length of up to 255 words located in certain address areas.

Operations

Table 9-4 Operations for field transfer

Operation	Operand	Function
TNW	0 to 255	Field transfer 0 to 255 words [1] (in the 16-bit memory area)
TXB	--	Field transfer from 8-bit to the 16-bit memory
TXW	--	Field transfer from the 16-bit to the 8-bit memory

[1] Can also be used for transfer from the byte to the byte area.

Parameters

Field length

For TNW: Operand = number of words (0 to 255)

For TXB/TXW ACCU 3 = number of words (0 to 127)

End address of the source area

ACCU 2 = end address of the source area (20 bits)

End address of the destination area

ACCU 1 = end address of the destination area (20 bits)

The **entire** source and destination areas must be located in one of the memory areas listed in Table 9-5 **and cannot overlap.**

Permitted memory areas

Table 9-5 Memory areas permitted for TNW, TXB and TXW

Addresses	Memory area
0 0000H to C FFFFH	User memory: 16-bit area (dependent on memory configuration)

Addresses	Memory area
Table 9-5 continued:	
E 8000H to E 9FFFH	System RAM: System data, 16 bits
E B000H to E FBFFH	System data (RI/RS, timers, counters etc.), 16 bits
E A000H to E AFFFH	S flags, 8 bits, low byte in 16-bit word (High byte not defined)
E FC00H to E FFFFH	Flags, process image, 8 bits (High byte = FFH)
F 0000H to F FFFFH	I/Os, 8/16 bits

(Refer also to Chapter 8)

Sequence

The field transfer is made in descending order, i.e. it begins with the highest address of the source area (= end address) and ends with the lowest.

TNW, TXB and TXW operations

The operations TNW, TXB and TXW are long-running STEP 5 operations which can only be interrupted by POWER DOWN and QVZ.

Special features

Interruptions by POWER DOWN

If one of the operations is interrupted by a power failure (NAU) followed by a warm restart, the operation does not resume at the point at which it was interrupted but from the **beginning** again.

Interruptions by QVZ

If a timeout (QVZ) occurs during the transfer, the operation is interrupted and the appropriate error OB called.
The error address is the address at which an error occurred (refer to Section 5.6.3).

ADF during execution

If an addressing error (ADF) occurs once or more than once during the transfer, all the part fields are first transferred and then OB 25 is called before the next operation is executed.

Example

TXB and TXW between 8 and 16-bit memory areas:

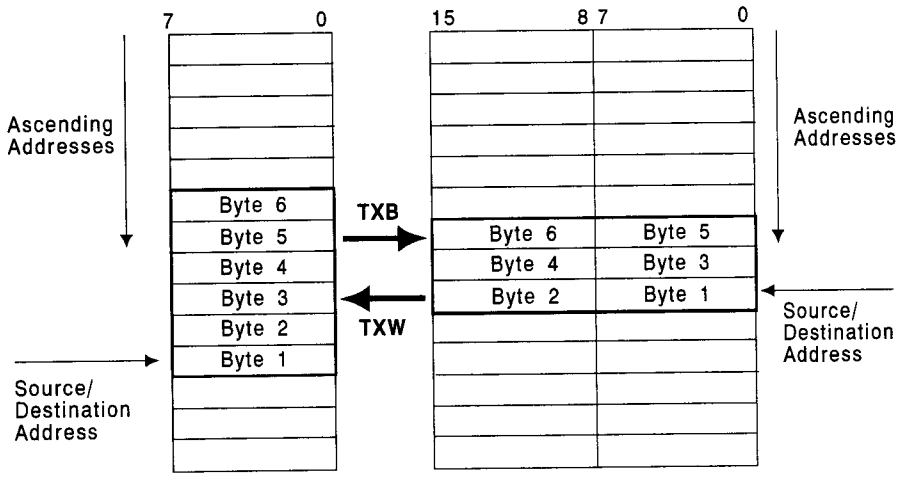


Fig. 9-7 Transferring memory fields

Transferring bytes 1 to 6 from an 8-bit to a 16-bit area:

```
:L    <field length in words> e.g.
:L    <source address>
:ENT
:L    <destination address>
:TXB
```

```
:L    KH 0003
:L    DH EFC10
:ENT
:L    DH EF208
:TXB
```

Transferring bytes 1 to 6 from a 16-bit to an 8-bit area:

```
:L    <field length in words> e.g.
:L    <source address>
:ENT
:L    <destination address>
:TXW
```

```
:L    KH 0003
:L    DH EF008
:ENT
:L    DH EFC10
:TXW
```

9.4 Operations with the Base Address Register (BR Register)

Application

The base address register (20 bits) allows you to calculate addresses and use indirect register load and transfer operations without using the ACCUs.
The memory location whose absolute address is calculated as the sum of the BR register content plus a constant is accessed.

Absolute address = BR register content + constant

Operations

Table 9-6 Load and arithmetic operations with the BR register

Operation	Operand	Function
MBR	Constant (0H to F FFFFH)	Load the BR register with a 20-bit constant
ABR	Constant (-32 768 to +32 767)	Add a 16-bit constant to the contents of the BR register

Changing the BR register

- The BR register is **retained**, when the program is continued in a **different block of the same program execution level** due to a jump statement ('JU FB'/'JC FB').
- The BR register is **retained** after **nesting in** a different program execution level.
- If a different program execution level is **called** by the system program, the BR register is set to **'0'**.

9.4.1 Operations for Transfer between Registers

Application

You can use the operations described in this section for the fast exchange of values between the registers ACCU 1 (32 bits), step address counter (SAC - 20 bits) and BR register (20 bits).

Operations

Table 9-7 Register-register operations

Operation	Operand	Function
MAS	--	Transfer the contents of ACCU 1 to the STEP address counter (20 bits)
MAB	--	Transfer the contents of ACCU 1 to the base address register (20 bits)
MSA	--	Transfer the contents of the STEP address counter (20 bits) to ACCU 1 [1)]
MSB	--	Transfer the contents of the STEP address counter (20 bits) to the base address register (20 bits)
MBA	--	Transfer the contents of the base address register (20 bits) to ACCU 1 [1)]
MBS	---	Transfer the contents of the base address register (20 bits) to the STEP address counter (20 bits)

[1)] Bits 2^{20} to 2^{31} are set to '0'.

The following figure illustrates how the registers are changed by the operations.

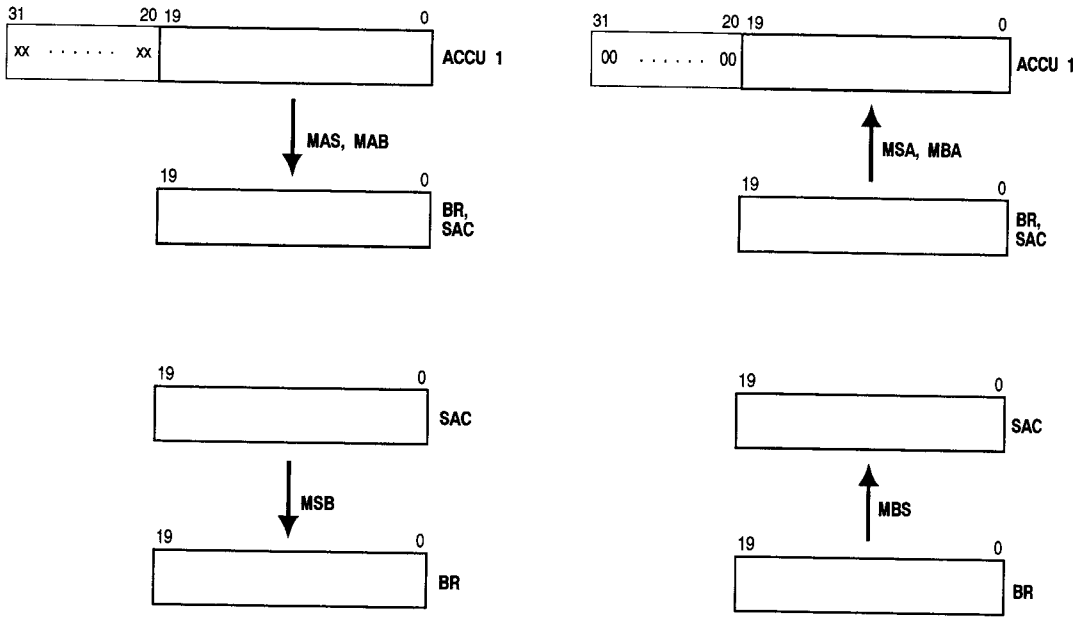


Fig. 9-8 Transfer operations from one register to another

9.4.2 Accessing the Local Memory

Application

The following operations allow access to the local memory organized in words using an absolute memory address. The absolute address is the sum of the contents of the BR register and the 16-bit constant contained in the operation (-32768 to +32767).

Operations

Table 9-8 Operations for accessing the local memory

Operation	Operand	Description
LRW	Constant (-32768 to +32767)	add the specified constant to content [1] of the BR register and load the word addressed in this way in ACCU-1-L
LRD	Constant (-32768 to +32767)	add the specified constant to content [1] of the BR register and load the double word addressed in this way in ACCU 1
TRW	Constant (-32768 to +32767)	add the specified constant to content of the BR register and transfer the content of ACCU-1-L to the word addressed in this way

Operation	Operand	Description
Table 9-8 continued:		
TRD	Constant (-32768 to +32767)	add the specified constant to content of the BR register and transfer the content of ACCU 1 to the double word addressed in this way

1) ACCU 2 $_{new}$ = ACCU 1$_{old}$

Error reaction

If the calculated address of the memory location is not between 0 0000H and E FFFFH, the CPU detects a load/transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the stop mode with the error code TRAF (ISTACK).

9.4.3 Accessing the Global Memory

Application

This section describes the operations you can use with absolute memory addresses to access the global memory organized in bytes or words. The absolute address is the sum of the contents of the BR register and the constants contained in the operation (-32768 to 32767).

Testing and setting an "occupied" register in the global area

You can control access of individual CPUs to commonly used memory areas by using an "occupied" register. An "occupied" register is assigned to each commonly used memory area. Each participating CPU must test this register before accessing the memory area. The "occupied" register contains either the value '0' or the slot ID of the CPU that is presently using the memory area. When the CPU is finished using the memory area, it **writes '0' to the "occupied" register to re-enable the memory area.** (Note the explanations for the operations "set semaphore/SED" and "enable semaphore/SEE" in Section 3.5.5.)

The TSG operation enables testing and setting of "occupied" registers.

Operation	Operand	Description
TSG	-32768 to +32767	Add the specified constant to the content of the BR register and test and set the location addressed in this way.

Sequence

The location used is the low byte of the word addressed by the BR register plus the constant. If the content of the low byte is '0', the TSG operation enters the slot ID in the location.

Testing (reading) and possible occupation of the location (writing) form a program unit that cannot be interrupted.

Result

You can evaluate the result of the test using condition codes CC 0 and CC 1:

CC 1	CC 0	Description
0	0	The "occupied" register contains '0'. The CPU enters its own slot ID.
1	0	The slot ID of the CPU is already entered in the "occupied" register.
0	1	The "occupied" register contains a different slot ID.

Note
All CPUs that require synchronized access to a **common global memory area** must use the TSG operation.

Error reaction

The absolute address must be between F 0000H and F FFFFH. If the absolute addresses are not in the range shown, the CPU detects a transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the STOP mode with the error code TRAF (ISTACK).

Load and transfer operations on the global memory organized in bytes

Table 9-9 Operations for access to the global memory organized in bytes

Operation	Operand	Description
LY GB	-32768 to +32767	add the specified constant to content of the BR register and load the byte addressed in this way in ACCU-1-LL [1) 3)]
LY GW	-32768 to +32767	add the specified constant to content of the BR register and load the word addressed in this way in ACCU-1-L [2) 3)]
LY GD	-32768 to +32767	add the specified constant to content of the BR register and load the double word addressed in this way in ACCU 1 [3)]
TY GB	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU-1-LL to the byte addressed in this way
TY GW	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU-1-L to the word addressed in this way
TY GD	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU 1 to the double word addressed in this way

1) ACCU-1-LH and ACCU-1-H are set to '0'.

2) ACCU-1-H is set to '0'.

3) ACCU 2 $_{new}$: = ACCU 1$_{old}$

Error reaction

The range of absolute addresses for the load and transfer operations for the global memory organized in bytes

- between F 0000H and F FFFFH (LY GB, TY GB),
- between F 0000H and F FFEH (LY GW, TY GW)

  or

- between F 0000H and F FFFCH (LY GD, TY GD).

9

If the absolute addresses are not in the range shown, the CPU detects a load/transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the STOP mode with the error code TRAF (ISTACK).

Load and transfer operations for the global memory organized in words

Table 9-10 Operations for access to the global memory organized in words

Operation	Operand	Description
LW GW	-32768 to +32767	add the specified constant to content of the BR register and load the word addressed in this way in ACCU-1-L [1) 2)]
LW GD	-32768 to +32767	add the specified constant to content of the BR register and load the double word addressed in this way in ACCU 1 [2)]
TW GW	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU-1-L to the word addressed in this way
TW GD	-32768 to +32767	add the specified constant to content of the BR register transfer the content of ACCU 1 to the double word addressed in this way

1) ACCU-1-H is set to '0'.

2) ACCU 2 $_{new}$: = ACCU 1$_{old}$

Error reaction

The range of absolute addresses must be located

- between F 0000H and F FFFFH (LW GW, TW GW)

  or

- between F 0000H and F FFFEH (LW GD, TW GD).

If the absolute addresses are not in the range shown, the CPU detects a load/transfer error (TRAF) and calls **OB 32**. ACCU 1 contains the error ID 1A01H. If OB 32 is not loaded, the CPU changes to the STOP mode with the error code TRAF (ISTACK).

9.4.4 Accessing the Page Memory

Application

With the following operations, you can access pages organized as **bytes or words** using an absolute memory address. The absolute address is the sum of the BR register content and the constant contained in the operation (-32768 to 32767).

Sequence of page access

Between the addresses F F400H to F FBFFH, the global memory area has a window for accessing one of a maximum of 256 memory areas called (pages). One page can occupy a maximum of 2K addresses and can be organized in either bytes or words. Before access to the page area, one of the 256 pages must be selected by entering its number in the **select register** (page address register). The procedure of writing to the select register and then accessing the page area cannot be interrupted.

Before you can access the page area (load/transfer), you must select one of the 256 dual-port RAM pages. Do this by putting the number of the page that you want to open into ACCU-1-L. Use the ACR operation to enter this number into the CPU-internal **page register.** All page operations that follow ACR write the contents of the page register into the select register of the corresponding modules on the S5 bus before page access.

Changing the page register

- The page register is **retained** when another block is called.
- If the page register is changed in a block, its value is **retained** when the program returns to the calling block at the end of the block.
- The page register is **retained** after nesting in another program execution level.

Opening a page

Operation	Parameter	Description
ACR		Open the dual-port RAM page whose number is located in ACCU-1-L , permissible values: 0 to 255

The page number must be between 0 and 255. If it is not, the CPU detects a substitution error (SUF) and calls **OB 27**. If OB 27 is not loaded, the CPU changes to the STOP mode.

Testing and setting an "occupied" register in the page area

The TSC operation handles the testing and setting of a location on the open page.

Operation	Operand	Description
TSC	-32 768 to +32 767	Add the specified constant to the content of the BR register and test and set the location addressed in this way on the open page.

Sequence

The low byte of the word addressed by the sum of the BR register + constant is used as the "occupied" register. If the low byte contains '0', the TSC operation enters the slot ID of the CPU in the "occupied" register.

Testing (reading) and possible occupation (writing) form a program unit that cannot be interrupted.

Result

You can evaluate the result of the TSC operation using condition codes CC 0 and CC 1:

CC 1	CC 0	Description
0	0	The "occupied" register contains '0'. The CPU enters its own slot ID.
1	0	The slot ID of the CPU is already entered in the "occupied" register.
0	1	The "occupied" register contains a different slot ID.

Error reaction

The location must be on the corresponding module and on the common page between F F400H and F FBFFH. If this is not the case, the CPU recognizes a transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the stop mode with the error code TRAF (ISTACK).

9

Load and transfer operations for the page memory organized in bytes

Table 9-11 Operations for access to pages organized in bytes

Operation	Operand	Description
LY CB	-32768 to +32767	add the specified constant to content of the BR register and load the byte addressed in this way in the open page into ACCU-1-LL [1) 3)]
LY CW	-32768 to +32767	add the specified constant to content of the BR register and load the word addressed in this way in the open page into ACCU-1-L [2) 3)]
LY CD	-32768 to +32767	add the specified constant to content of the BR register and load the double word addressed in this way in the open page into ACCU 1 [3)]

Operation	Operand	Description
Table 9-11 continued:		
TY CB	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU-1-LL to the byte addressed in this way in the open page
TY CW	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU-1-L to the word addressed in this way in the open page
TY CD	-32768 to +32767	add the specified constant to content of the BR register and transfer the content of ACCU 1 to the double word addressed in this way in the open page.

1) ACCU-1-LH and ACCU-1-H are set to '0'.

2) ACCU-1-H is set to '0'.

3) ACCU 2 $_{new}$: = ACCU 1$_{old}$

Error reaction

The range of absolute addresses must be

- between F F400H and F FBFFH (LY CB, TY CB),
- between F F400H and F FBFEH (LY CW, TY CW)

  or

- between F F400H and F FBFCH (LY CD, TY CD).

If the absolute addresses are not in the range shown, the CPU detects a load/transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the STOP mode with the error bit TRAF (ISTACK).

Load and transfer operations for the page memoryorganized in words

Table 9-12 Operations for access to pages organized as words

Operation	Operand	Description
LW CW	-32768 to +32767	add the specified constant to content of the BR register and load the word addressed in this way in the open page into ACCU-1-L [1]
LW CD	-32768 to +32767	add the specified constant to content of the BR register and load the double word in the page addessed in this way in the open page into ACCU 1 [2]
TW CW	-32768 to +32767	add the specified constant to content of the BR register and transfer the contents of ACCU-1-L to the word in the open page addressed in this way.
TW CD	-32768 to +32767	add the specified constant to content of the BR register and transfer the contents of ACCU 1 to the double word in the open page addressed in this way.

[1] ACCU-1-H is set to '0'.

[2] ACCU 2 $_{new}$: = ACCU 1$_{old}$

9

Error reaction

The range of absolute addresses must be

- between F F400H and F FBFFH (LW CW, TW CW)

  or

- between F F400H and F FBFEH (LW CD, TW CD).

If the absolute addresses are not in the range shown, the system program detects a load/transfer error (TRAF) and calls **OB 32**. If OB 32 is not loaded, the CPU changes to the STOP mode with the error bit TRAF (ISTACK).

PG Interfaces and Functions 10

Contents of Chapter 10

10.1 Overview ... 10 - 4

10.2 Notes on Connecting the PG and the PLC Modes ... 10 - 5

10.2.1 Connecting the PG ... 10 - 5
10.2.2 PLC Modes ... 10 - 6

10.3 PG Functions ... 10 - 8

10.3.1 Info ... 10 - 9
Memory configuration ... 10 - 9
Output DIR ... 10 - 10
Output address ... 10 - 10
10.3.2 Installation ... 10 - 10
Overall reset ... 10 - 10
Compress memory ... 10 - 10
Transfer block ... 10 - 11
Delete block ... 10 - 11
10.3.3 Program Test ... 10 - 11
Start/stop ... 10 - 11
Status block ... 10 - 12
Program test ... 10 - 13
Status variables ... 10 - 16
Force ... 10 - 16
Force variables ... 10 - 17

-

10.4 Serial Link PG - PLC via the Serial Interface ..... 10 - 18

10.5 PG Functions via the S5 Bus ..... 10 - 19

10.5.1 Application ..... 10 - 19
10.5.2 How the PG Functions via the S5 Bus Work ..... 10 - 20
10.5.3 General Notes on Installation ..... 10 - 21
10.5.4 Installation with One-Sided Operation ..... 10 - 22
The CP 143 is only used for PG functions ..... 10 - 23
The CP 143 is used for PG functions and communication SINEC H1 ..... 10 - 24
10.5.5 Installation for Switched Operation ..... 10 - 26
Configurations ..... 10 - 26
Installation ..... 10 - 29
Special features ..... 10 - 30
Specifying and editing bus paths ..... 10 - 30
10.5.6 Displaying Problems ..... 10 - 31

PG Interfaces and Functions

10

This chapter explains how to connect your PG to the CPU 948 and the functions provided by the PG software with which you can test your STEP 5 program.
If you only use the standard PG interface (1st serial PG interface) you do not need to read Section 10.5. This section tells how to connect a PG to your CPU via the S5 bus with SINEC H1. This section also contains points to note if you use PG functions on both interfaces.

10.1 Overview

You can load and test your user program using the online functions of the STEP 5 software.

To use these functions, the CPU must be connected to the PG. The following interfaces are available for this link:

- link via the serial standard interface "PG - PLC",
- link via the S5 bus with SINEC H1.

The PG functions via the S5 bus with SINEC H1 can only be used **alternately** with those on the serial interface.

PG functions provide the following support for installing and testing your STEP 5 program:

Table 10-1 Functions for installation and testing

Function	Section
Info	
Size of the internal RAM and free user memory	"Memory configuration"
List of loaded blocks	"Output DIR"
Display contents of memory words/bytes and I/O bytes	"Output address"
Memory management	
Delete the whole memory	"Overall reset"
Create more memory space	"Compress memory
Manage blocks	"Transfer/delete blocks"
Program test	
Start/stop CPU	"Start/stop"
Test the operation sequence in a block	"Status block"
Test single program steps	"Program test"
Display signal state of process variables	"Status variables"
Output signals in the stop mode	"Force"
Display/change process variables	"Force variables"

Note
The PG functions extend the cycle time required

10.2 Notes on Connecting the PG and the PLC Modes

10.2.1 Connecting the PG

- If a PG function is interrupted by the cable between the PG and PLC being disconnected or by an error in the processing of a function (timeout etc.) it can take up to 30 sec. before access to the PLC is possible again. Requests for functions during this time are rejected with the message -"AS function disabled: function active".

- In the redundant mode, with longer cycle times (greater than 500 ms), some functions called with the LAD, CSF, STL package cause the monitoring time set on the PG to be exceeded (PLC timeout). **Remedy**: switch on timed interrupts in the DX 0 screen form.

10.2.2 PLC Modes

Remember that the S5-155H has four modes:

1. Solo mode (only the master in RUN),
2. Coupling standby,
3. Redundant mode,
4. Error search mode (error search is only performed by the standby; during this time, the master is in the solo mode). In terms of the PG functions, this mode has the same effect as the coupling standby mode.

Working at the PG during the solo and coupling standby modes

Solo mode

PG online with standby PLC:



Caution
If programs are modified on a standby PLC in the STOP mode, the changes are lost as soon as the standby PLC is connected to the master!

Coupling standby

During the coupling standby phase, all write access "PG – PLC" both to the master and to the standby is rejected with the error message "Wrong mode at PLC". The status "coupling standby" is not directly recognizable on the PG.

Working at the PG during the redundant mode

General information about the PG functions

While the redundant mode is active, both PLCs are in the RUN mode. PG functions are possible with both PLCs. The master and standby are ranked equally.

- All **read functions** in the redundant mode are the same as with the CPU 948. Only the data of the connected PLC are output. This means that with the functions "**Status" of formal operands on F page** (F 0000 to F FFFF) and "**Output address" on F page**, no valid values are indicated at the standby PLC which cannot access the switched I/Os.

- The effects of all **write functions** in the redundant mode are extended compared with those of the CPU 948. The data entered are transferred to the partner PLC at the cycle checkpoint. All write functions therefore take longer than with the U system.

Working with subunit A and subunit B simultaneously

If a programmer is connected to both subunits, remember the following points :

- Read functions on one side are generally possible at the same time as **write functions** on the partner.

- "Output address" (read) at the same time as, for example "Input block" (write) is, however, pointless and is therefore not permitted.

- Write functions are permitted on both sides. If the write functions are simultaneous, only the **first to be recognized** is executed. The second is rejected with the error message "AS function disabled: function active".

10.3 PG Functions

> **Note**
> The terms used in this section for the PG functions may, in some cases, differ from the terms in your PG software.
> Please refer to your STEP 5 manual.

Calling and using functions

How to call and use the individual PG functions is described in the manual for your PG.

Checkpoint

The PG functions are performed at defined **checkpoints** in the CPU. In the CPU 948R there are four different checkpoints. Each of these checkpoints has certain test functions assigned to it.

Checkpoint "Stop"
You can access PG functions that are permissible only in the STOP mode at checkpoint "stop" (e.g. "start", "overall reset", "compress memory" in the stop mode).
The "stop" checkpoint is located immediately **before** OB 39 is called (the block for cyclic processing in the soft STOP mode).

Checkpoint "Cycle"
PG functions that you want to execute during cyclic program processing are called at checkpoint "cycle" (e.g. "compress memory" in the RUN mode, "stop", "status"). The "cycle" checkpoint is located immediately **before** the updating of the process image of the inputs (PII). At this point, the system program has not yet updated the PII.

Checkpoint "Test"
PG functions that you want to execute as soon as the next breakpoint is reached are called at checkpoint "test" (in the program test, see Section 10.2.3).

Checkpoint "General Functions"
This checkpoint exists both in the STOP and the RUN mode. Online PG functions that can be executed in all operating modes of the programmable controller are called at checkpoint "general functions". These functions include "transfer block", "delete block", "status variables". In the STOP mode, this checkpoint is located immediately **before** OB 39 is called (the block for cyclic processing in the soft STOP mode). During cyclic program processing, the checkpoint is located **after** updating the process image of the inputs and before calling OB 1 (block for cyclic program execution in RUN).

10.3.1 Info

Memory configuration

The CPU 948R is available with two memory versions and you can check the memory capacity using the PG function "memory configuration". With this function, the following information about the CPU user memory is transferred to the PG (from PG software version STEP 5/ST V6.3 upwards or STEP 5/MT V6.0 both with "Delta diskette" for the CPU 948):.

- memory capacity (640 Kbytes/1664 Kbytes)
- longest free block of user memory
- sum of all free blocks of user memory.

The PG software mentioned above displays the memory configuration exactly.
With older PG software versions (e.g. V3.0 or MT1.0) the memory configuration is as for the CPU 946R/947R (refer to Fig. 10-1). The total memory configuration of the CPU 948 must then be calculated from the total of the submodule values.

PLC info — SIMATIC S5 / OESOC

Memory configuration S 5 1 5 5 U

Module	Submod	Type	Start address	End address	Length	
0	1	RAM	00000	0FFFF	64 KW	
1	1	RAM	10000	2FFFF	128 KW	1)
1	2	RAM	30000	4FFFF	128 KW	
1	3		Submod. empty / not plugged in			
2	1		Submod. empty / not plugged in			
2	2		Submod. empty / not plugged in			
2	3		Submod. empty / not plugged in			

Longest free block of RAM : 314800 Words
Sum of all free blocks of RAM : 315664 Words

F1	F2	F3	F4	F5	F6	F7	F8
OUTP ADDR	MEM CONF	SYSPAR	BSTACK	ISTACK			RETURN

1) corresponds to the memory configuration of the CPU 948R-1: 320 Kw = 640 Kbytes

Fig. 10-1 PG display of the memory configuration with older PG software

Output DIR

The function is permitted in the operating modes RUN, SOFT STOP, HARD STOP and can also be called within the "program test" function.

Output address

With the "output address" function, you can display the contents of memory and I/O addresses in hexadecimal format. You can access all addresses (RAM, S5 bus, areas with no modules assigned). In the process image area no ADF is triggered, in the I/O area there is no QVZ.

In the areas addressed as bytes (flags, process image) the high byte is represented as 'FF'.

10.3.2 Installation

Overall reset

With the function "delete all blocks" you can carry out an overall reset of the CPU from the PG. The overall reset is carried out unconditionally (refer to Section 4.2).

> **Note**
> The "delete all blocks" function is not permitted in the RUN mode. If you attempt to execute the function, it will be rejected with the message "Wrong mode at PLC" or "Memory error on PLC".

Compress memory

The function is permitted in the solo and redundant modes.

This function shifts all valid blocks in the user memory to the beginning of the user memory. Unused areas that resulted from deleting or correcting blocks are eliminated. This function shifts complete blocks to the beginning of the memory area. Ideally, one large free area results from many small unused areas. You can load blocks into the resulting large space.

You can call this function in the RUN and soft STOP modes. In the RUN mode, DBs and DXs that are longer than 512 data words are **not** shifted. In the STOP mode, **all** blocks are shifted.

The blocks are shifted via a buffer so that no data is lost if there is a power failure. If this buffer is insufficient for intermediate storage of a block, compressing continues at the next unused memory area. Consequently, some unused areas can still remain after compressing.

See also Section 8.3.

Transfer block

With this function you can transfer new or existing logic and data blocks to the user memory of the CPU.

If a block already exists in the user memory of the CPU, it is declared invalid and the new block becomes valid.

Restrictions when transferring from FD

If you want to transfer blocks from FD to PLC, remember the following restrictions:

1: "TRANSFER SOURCE : FD BLOCK [DX1] TO DEST: PC"

This function is not permitted in RUN. if you attempt this, the system message "Block type/no. illegal" is displayed.

2. "TRANSFER SOURCE : FD BLOCK [RAM-DB] TO DEST: PC"

The RAM-DB must not be transferred in RUN ! If you attempt this, the system message "DMA interface not ready" or "Timeout" is displayed.

Delete block

With this function you declare a logic or data block in the user memory as invalid.

The space taken up by such blocks is released and can be used again.

10.3.3 Program Test

Start/stop

These functions on the PG correspond to manual operation.

The "Start" function is only effective on the subunit to which your PG is connected.

With the "Stop" PG function in the RUN mode, you stop the subunit to which the PG is connected.

The COM H functions "PLC START" and "PLC STOP" affect either subunit A or B as required or both subunits.

You exit the SOFT STOP status with a COLD RESTART or COLD RESTART WITH MEMORY. The CPU exits the stop mode.

Status block

You can call the "status" PG function to test related operational sequences (STEP 5 operations) in one block at any location in the user program.
The current signal status of operands, the accumulator contents, and the RLO are output on the PG screen for every executed operation in the block (i.e. step mode). You can also use this function to test the parameter assignment of function blocks (i.e. field operation):
The signal status of the actual operands is displayed.

If the actual operands involve direct access to the I/Os then

- with one-sided I/Os, the actual values are only displayed for the subunit in which the I/Os are plugged in,
- with switched I/Os only the actual values for the master are displayed.

The "status block" function (with correction) may extend the time for the master-standby switchover.

Calling the function and specifying a breakpoint

When you call the "status" function on a PG and enter the type and number of the block you want to test (possibly including the nesting sequence and search key), you enter a breakpoint.

When the "status" function is called during program processing in the RUN mode, program processing continues until it reaches the operation marked by the specified breakpoint in the correct nesting sequence. Then the system program executes each of the monitored operations up to the operation boundary, outputting the processing results to the PG.

> **Note**
> The results of operation processing are not output in each of the subsequent cycles.

Nesting and interruptions

A sequence of operations marked by a breakpoint is completed even if a different program execution level (e.g. an error OB or interrupt OB) is activated and processed. With this you can see whether data has been changed by nested program sections.

If an interruption in a nested program execution level puts the CPU into the STOP mode, data is output up to the operation that was executed before the program execution levels changed. The data of the remaining operations is padded with zeros (the SAC is also 0).

The "status" function is possible in the following modes. RUN, RESTART (OB 20, OB 21, OB 22) soft STOP (OB 39 **only**).

Program test

The function is not permitted in the redundant mode.

You can call the "program test" function to test individual program steps anywhere in your user program. When you do this, you stop program processing and allow the CPU to process one operation after the other. The PG outputs the current signal status of operands, the accumulator contents, and the RLO for each operation executed.

Calling the program test and specifying the first breakpoint

You can call the "program test" function in the RUN and soft STOP modes. To call the function, specify the **type** and **number** of the block you want to test. You may also want to include the nesting sequence. At the PG, mark the first operation you want to test. This is how you specify the first breakpoint.

> **Note**
> You must not select a substitution operation as the breakpoint with DO FW/DO DW otherwise the function is aborted and the CPU changes to the cyclic mode.

When you specify the first breakpoint during **program processing**, the CPU continues processing the program until it reaches the operation marked by the specified breakpoint. The operation is executed up to the operation boundary. (The DO FW and DO DW operations are processed **including** the substituted operation.) The CPU checks to see if the current block nesting sequence matches the block nesting sequence that you specified. If the nesting sequences do not match, the CPU continues program processing.

If program processing does not reach the specified breakpoint (e.g., because the CPU goes into the STOP mode or there is a continuous loop in the user program), the PG displays the message "Statement not processed". However, the function and the specified stopping point remain active.

If the nesting depths match, the output command is disabled (the "BASP" LED is on) and the PG displays the data of the processed operation. The CPU waits for further instructions from the PG.

Calling test functions in SOFT STOP

You can also call the "program test" function and specify an initial breakpoint when the CPU is in the soft STOP mode. The CPU remains in the soft STOP mode, and you can execute either a COLD RESTART or a MANUAL WARM RESTART. The CPU processes the program up to the marked operation and it proceeds as outlined above.

Executing the function and specifying another breakpoint

Initial situation: the CPU has processed the 1st break point.

To continue the function, you have two possibilities:

1. Specify the next operation as the **following** breakpoint:

   Move the cursor down to the next operation to specify the following breakpoint.
   The CPU continues by processing this operation up to the operation boundary. Then the CPU outputs the data and waits for further instructions from the PG. However, if a nested program execution level interrupts operation processing at the following breakpoint, the CPU processes the nested program first. Then the CPU returns to the 2nd breakpoint that you specified.

> **Note**
> You **cannot** specify a following breakpoint when the CPU is in the STOP mode.

2. Specify a **new** breakpoint:

   At the PG, specify any other operation in the same block or in a different block. The CPU continues program processing until it reaches the new breakpoint. The CPU processes the operation up to the operation boundary, then it outputs the data.

Cancelling the breakpoint

You can cancel the breakpoint by pressing the <BREAK> key if the CPU has not reached this breakpoint. After that you can specify a new breakpoint or terminate the program test.

Aborting the function

You can abort the "program test" function during program processing and when the CPU is in the soft STOP mode by calling the "program test end" function. The CPU goes into the STOP mode (or stays in STOP). The STOP-LED flashes slowly. BEARBE is marked in the control bits display. Afterwards a COLD RESTART is required.
The function is also aborted if an interface error occurs during the "program test" function (i.e., the cable between the PG and the programmable controller is disconnected).

Interruptions

- Program processing → STOP mode:

  If an interruption occurs during program processing (e.g., STOP switch, error OB not programmed etc.) before the program reaches the specified breakpoint, the CPU goes into the STOP mode immediately. If you execute a COLD RESTART or a MANUAL COLD RESTART WITH MEMORY, the "program test" function is still in effect and the breakpoint is still set.

- Program processing at breakpoint → STOP mode:

  If stop conditions occur at the breakpoint or next breakpoint during program processing, the CPU goes directly into the soft STOP mode and outputs the data.
  If you do not specify a new breakpoint while the CPU is in the STOP mode, the "program test" function is still in effect after the restart.

While the "program test" function is in effect, you can execute the following other functions at your PG

- Output ISTACK
- Output BSTACK
- Load block
- Read block
- Delete block
- Output block list
- Force variables
- Force

Status variables

Using the "status variables" function, you can display the current signal states of certain operands (process variables).

When a checkpoint is reached, the PG displays the present signal status of the desired process variable. You can display the following process variables: inputs, outputs, flags, timers, counters and data words. No addressing error (ADF) is triggered in the process image area.

*Sequence in **RUN***

If the function is active in the RUN mode, the signal states of the operands are scanned and displayed when the checkpoint is reached. The system program reads inputs from the **process image**. As long as the function is not aborted, signal states are updated cyclically.

> **Note**
> If the program does not reach the system checkpoint, the system program does not output the signal states (e.g., in a continuous loop in the user program).

*Sequence in **SOFT STOP***

When the function is active in the STOP mode, the signal states of the operands are displayed when they exist at the system checkpoint. It is important to note that the inputs are scanned and output directly on the I/O module.

Force

You can call the "force" PG function to manually set the output bytes of the programmable controller to the signal states you want.

> **Note**
> The "force" function is only permitted in the **stop mode** (SOFT STOP mode or within the "program test" function).
>
> With the CPU 948R, the "force" function is possible with switched I/Os providing the system configuration (DX 1) is loaded and both PLCs are in the STOP mode.

Sequence of the function

When you call the "force" function in the STOP mode, the disable output command signal is cancelled (i.e., the BASP LED is off). **All** digital peripherals are cleared (i.e., the value 0 is written to each address). While the peripherals are being cleared, this function cannot be interrupted. If any timeout signals (QVZs) occur while the outputs are being cleared, they are ignored.

The peripheral outputs are forced byte by byte.

Timeout errors that occur are detected when outputs are changed (PG message "I/O module does not exist").

Terminating the function

You can terminate the function by pressing the <BREAK> key on the PG. The disable output command signal is active again (i.e., the "BASP" LED is on). The function also ends if the CPU goes into the RUN mode between calling the function and actually forcing outputs.

Force variables

You can call the "force variables" function to look at the values of operands (process variables) in the process image table and change them. You can use this function in the RUN and soft STOP modes and within the "program test" function. You can display the following process variables: inputs, outputs, flags, timers, counters and data words.

Special features

Any change becomes effective at the next system checkpoint, i.e. regardless of the system checkpoint (start of cycle or end).

Note that the forced values can be overwritten again (e.g. by the user program or by the process image updating).

Note
The PG forces process variables in bytes.

If you are forcing **several operands**, the bytes are changed in memory one after the other distributed over several cycles.

10.4 Serial Link PG - PLC via the Serial Interface

For the serial link PG - PLC there are the following possibilities:

- Direct link to the CPU via the standard cable.
- Link to the PG via a PG multiplexer 757. The permitted cables can be found in the system manual 135U/155U /2/.
- Link to the PG via SINEC H1/L2/L1 and "swing cable"; the COR C or PG multiplexer can be connected in the link.

10.5 PG Functions via the S5 Bus

10.5.1 Application

The PG functions via the S5 bus with SINEC H1 S5-155H programmable controllers with the CPU 948R networked via SINEC H1 to be loaded and controlled from the PG 7xx. Using the PG functions via the S5 bus the CPU 948R can be loaded in the STOP mode up to eight times faster than via the PG interface. The speed depends on the length of the blocks to be transferred.

The PG functions via the S5 bus are part of the system program of the CPU 948R.



Caution
The PG functions via the S5 bus can **only** be used **alternately** with the PG functions via the serial interface.
On the CPU 948R, interface numbers ≥ 232 are reserved for the PG functions via the S5 bus. These numbers cannot be used freely for handling blocks.

Assigning parameters to the CPU

Before you can use the PG functions via the S5 bus in the RUN mode, you must first configure the system (CP interfaces, switched EUs ...) with COM 155H in data block DX 1 and transfer the data block to the PLC.

Technical requirements

The PG functions via the S5 bus with the CPU 948R require that the PG and PLC are networked via SINEC H1.

The following is necessary:

- a PG 7xx with a SINEC H1 connection and with STEP 5 software version 6.3 (ST) or 6.0 (MT) + delta diskette CPU 948 installed.
- in the PLC (central controller or expansion unit EU 185) there must be a communications processor CP 143 version 06 or higher (firmware version 3.0) with selected base interface number 232 or 236 (the base interface number is set on the hardware with jumpers and in the SYSID with COM 143).

Note
For the PG functions via the S5 bus, CP 143 switched operation must not be mixed with one-sided CP 143 operation.

10.5.2 How the PG Functions via the S5 Bus Work

Using pages

With the S5 bus functions, the pages of the CP are divided into two pages for user HDBs and two pages for PG functions.
The pages for user HDBs can be used as normal for SINEC H1 applications. Remember, however, the special points listed in Section 10.5.4 .
The pages for PG functions are used by the CP 143 and the CPU 948R for PG functions via the S5 bus and are therefore no longer available for communication via handling blocks.

Interface numbers (SSNR)

The PG functions via the S5 bus are activated on the CP 143 automatically, when you set **base interface number 232 or 236** on the CP (jumpers **and** SYSID!). You then occupy interface numbers 232 to 239. The interface numbers 240 to 247 are reserved for later expansions (e.g. CPs with 8 pages/interfaces).

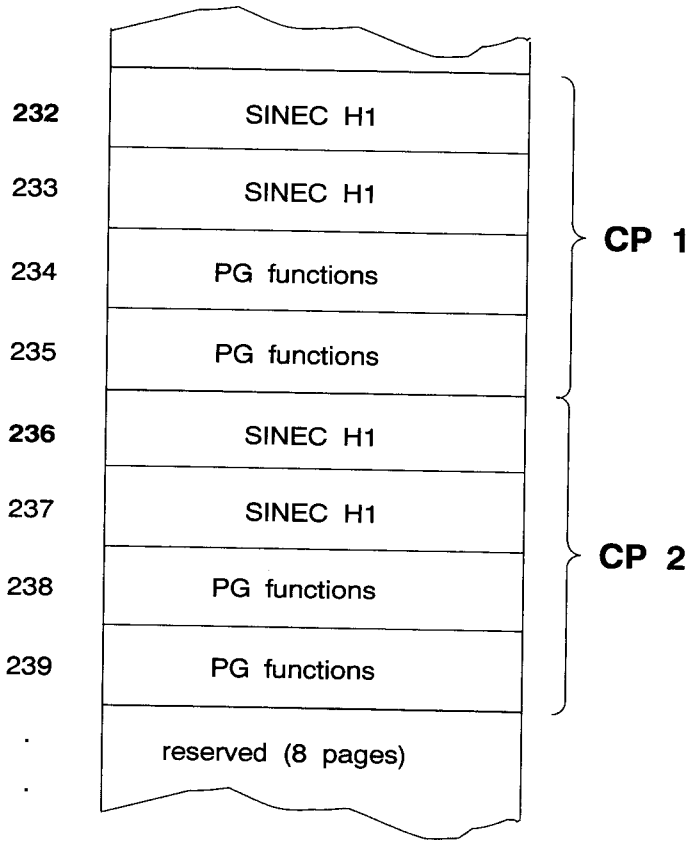


Fig. 10-2 Interface assignment of the PG functions via the S5 bus

Assigning parameters to the CP 143

Parameter assignment for the CP 143 is described in the CP 143 manual /6/.



Caution
The interface numbers 232ff and 236ff must not be used when the CP 143 is operating with other SIMATIC CPUs. When using the CPU 948R with other CPs interface numbers 232 to 247 must not be used.

10.5.3 General Notes on Installation

Installing the link

Installing the CP 141x in the PG 7xx

This step is only necessary when you add the CP 141x to your PG 7xx at a later point in time.

Installing the CP 141x module (including parameter assignment) is described in the Product Information for the CP 141x, which is supplied with the module.

Installing the CP 143 in the S5-155H

Installing the CP 143 is described in Part I of Volume 1 of this manual under "CP 143 with COM 143".

The slots permitted for installation of the CP 143 in the S5-155H can also be found in Part I of Volume 1 of this manual.

Connecting the PG 7xx and CP 143

Connect the CP 143 and the PG 7xx with th cable supplied with the PG 7xx.

10

Points to note

- At any one time, a subunit of the CPU 948R processes only one PG function. If you attempt to activate a second function from a second PG via the serial PG interface while a PG is running, the message "AS function disabled: function active" is displayed on the second PG..
- If you abort a PG function by changing the mode selector on the CPU from RUN to STOP or if an error occurs, this results in waiting times of longer than 15 sec. in the communication via SINEC H1.
- If you make a mistake on the PG (e.g. for example you switch off the PG while a function is active, it may be necessary to repeat the "Selection" function when the path is re-established.

10.5.4 Installation with One-Sided Operation

The Figures 10-3 and 10-4 illustrate two possible configurations with a one-sided CP 143

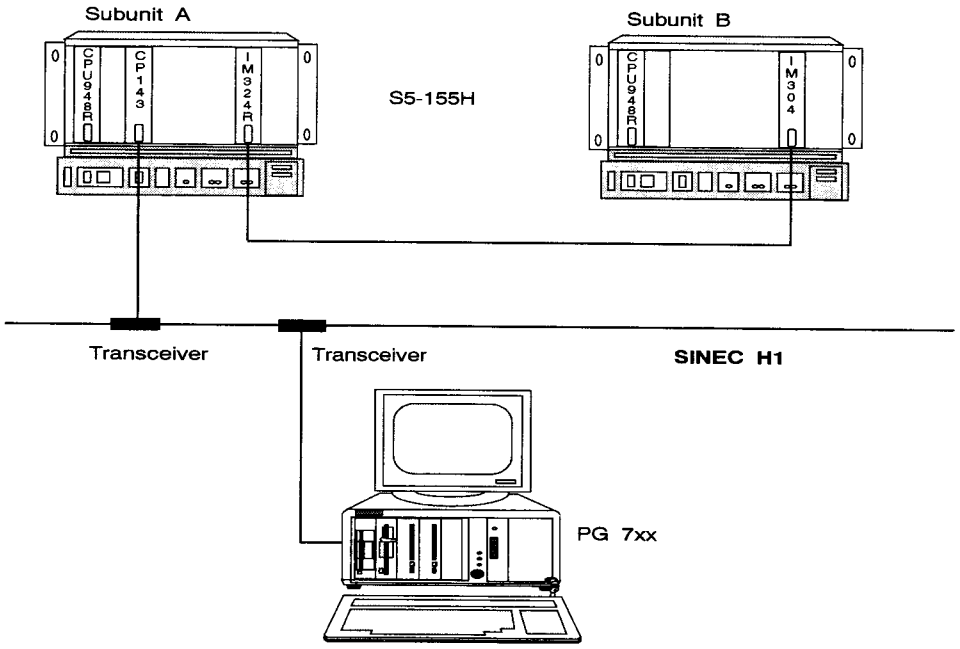


Fig. 10-3 One-sided operation, configuration 1: with **one** one-sided CP 143

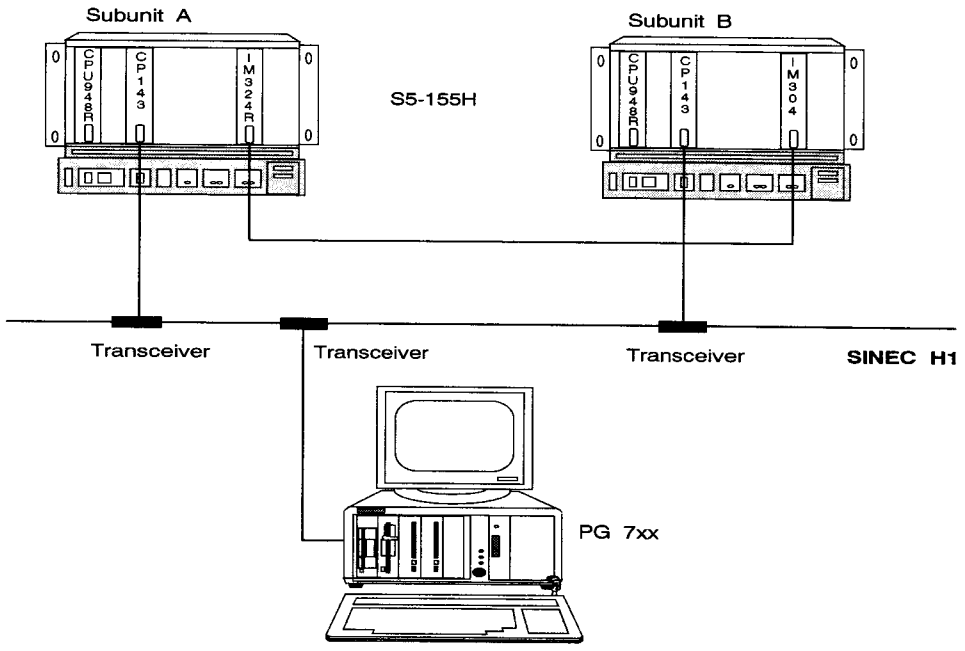


Fig. 10-4 One-sided operation, configuration 2: with **two** one-sided CP 143s

During installation, remember the two following alternatives:

The CP 143 is only used for PG functions

After switching the POWER ON; the PG functions via the S5 bus on the CPU 948R are always available without the CP 143 being previously synchronized using HDB SYNCHRON (FB 125).
An "empty" CPU 948R can be started up via the S5 bus without an OVERALL RESET.

The CPU 948R synchronizes the pages for PG functions via the S5 bus on the CP 143 automatically after POWER UP.

The following steps are necessary:

<table>
<tr><th>Step</th><th colspan="2">Action</th></tr>
<tr><td>1</td><td colspan="2">Set the interface number (SSNR) on the CP 143 (jumpers):
Select the SSNR according the existing hardware configuration as described below.
Remember the instructions in /6/.</td></tr>
<tr><td></td><td>Possible hardware configuration</td><td>Corresponding SSNR of the CP 143(s)</td></tr>
<tr><td></td><td>Configuration 1</td><td>232 or 236</td></tr>
<tr><td></td><td>Configuration 2</td><td>232 on 1st CP, 236 on 2nd CP</td></tr>
<tr><td>2</td><td colspan="2">Plug the CP 143 into the S5-155H (the power to the PLC must be off!).</td></tr>
<tr><td>3</td><td colspan="2">Connect the PG to the PG interface of the CP 143 and load COM 143.</td></tr>
<tr><td>4</td><td colspan="2">Set the same hardware interface number as in step 1 and the Ethernet address in the SYSID of the CP 143 with COM 143.</td></tr>
<tr><td>5</td><td colspan="2">Load the parameter data on the CP 143:
The parameter data of the CP 143 can be stored
- in an EPROM
or
- in the RAM of the CP 143.

You can transfer the parameter data via the serial interface of the PG 7xx. The individual steps for loading the parameter data of the CP 143 can be found in /6/.</td></tr>
<tr><td>6</td><td colspan="2">Switch the power for the PLC on again.</td></tr>
<tr><td>7</td><td colspan="2">Edit the path to the CPU 948R in the STEP 5 bus selection screen.</td></tr>
<tr><td>8</td><td colspan="2">Select the path via SINEC H1/CP 143 to the CPU 948R in the STEP 5 presets screen.</td></tr>
</table>

The PG functions via the S5 bus can be used after these steps have been taken. You can now load and run or test your user program.

10

The CP 143 is used for PG functions and communication via SINEC H1

If you want to use the CP 143 for communication via SINEC H1 as well as for the PG functions via the S5 bus, you must make further settings in addition to those described above and must take into account further special features.

When installing, follow the procedure outlined below:

Step	Action
1 to 7	Steps 1 to 7 are identical to those when the CP 143 is used only for PG functions.
8	**In the start-up OBs, OB 20 and OB 22, program the call for the HDB SYNCHRON (FB 125), so that during MANUAL/AUTOMATIC COLD RESTART and AUTOMATIC COLD RESTART WITH MEMORY, the CP 143 is synchronized for SINEC H1 communication.**

Pages for communication via handling blocks

Fig.10-5 illustrates how the PG functions via the S5 bus occupy the pages of the CP 143. The free pages for user HDBs can be used by the CPUs "subunit A" and "subunit B" for communication via SINEC H1.

Special features

In the following situations, the link between the PG and CPU must be re-established:

- After a change from STOP to RUN
  (Owing to this reaction, you cannot use the PG function "**status block**" via the S5 bus for **start-up OBs**
- after depassivation
- after a change to the redundant mode.

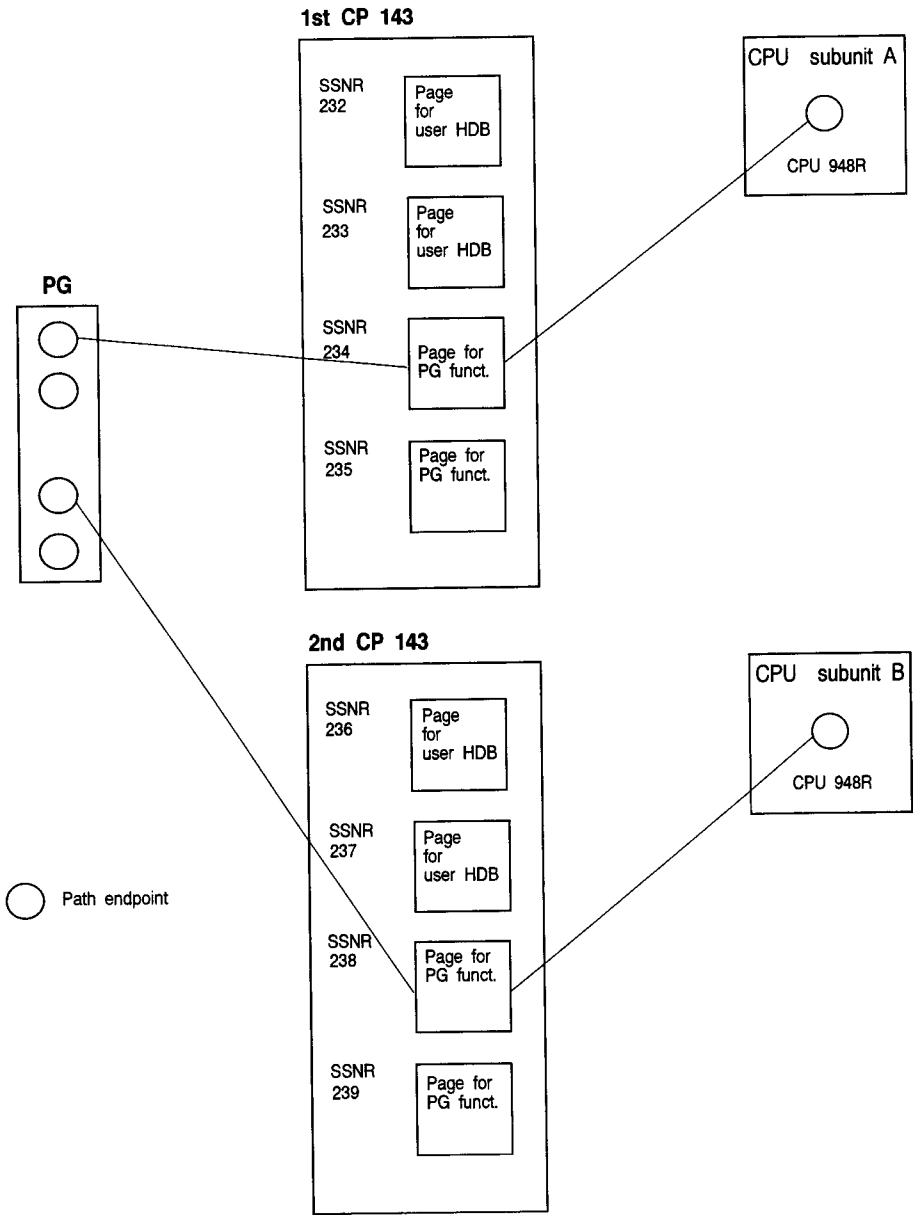


Fig. 10-5 Paths between PG and CPU 948R and assignment of the pages of the CP 143 in one-sided operation

Note
When you start up with COM 155H, the connected PLC automatically becomes master; the standby PLC then automatically accepts the user program during START-UP.

10

10.5.5 Installation for Switched Operation

Configurations

Each switched EU has two IM 314R (bus converters). The bus converters connect the master PLC to the EU directly.

The EUs must be set to EU numbers '14' and/or '15' using the DIP switches on the IM 314R. You must also configure the EUs and CPs in data block DX 1 with COM 155H.

The following three figures illustrate possible configurations of the PG functions via the S5 bus with switched expansion units.

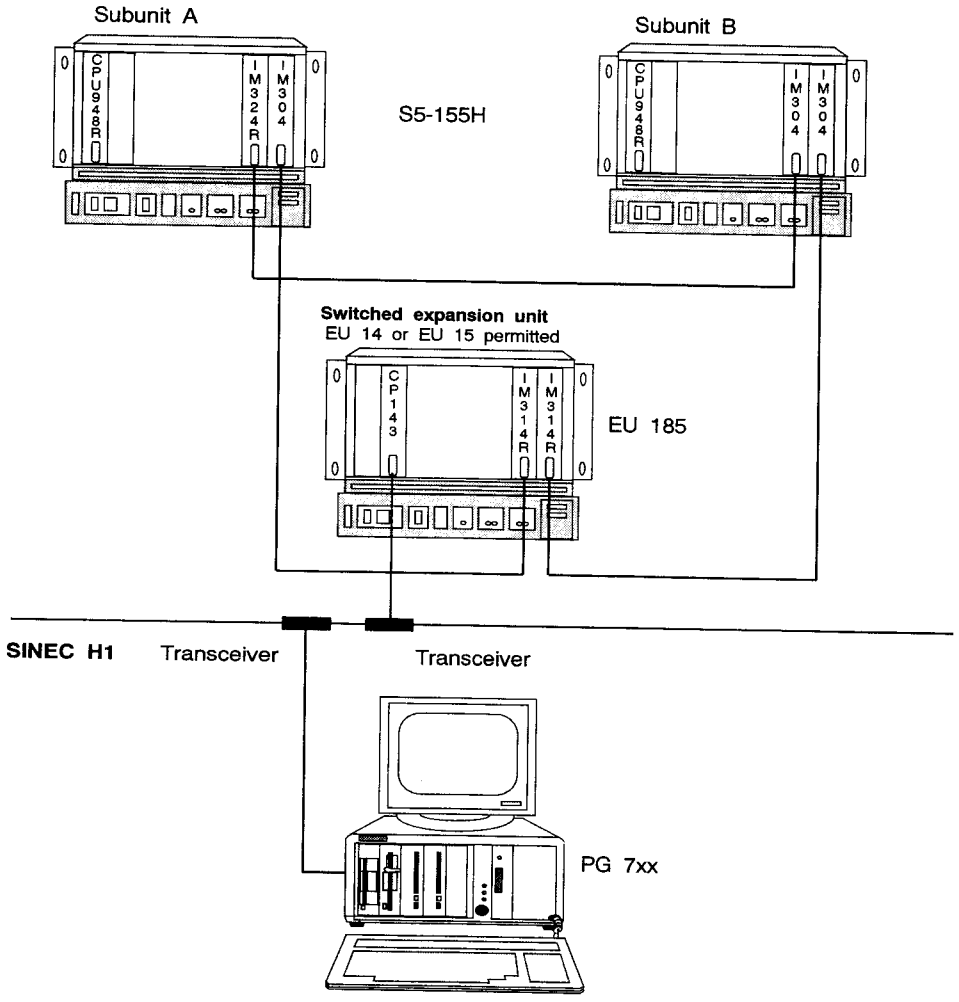


Fig. 10-6 Basic configuration with switched EU: operation with **one** CP 143

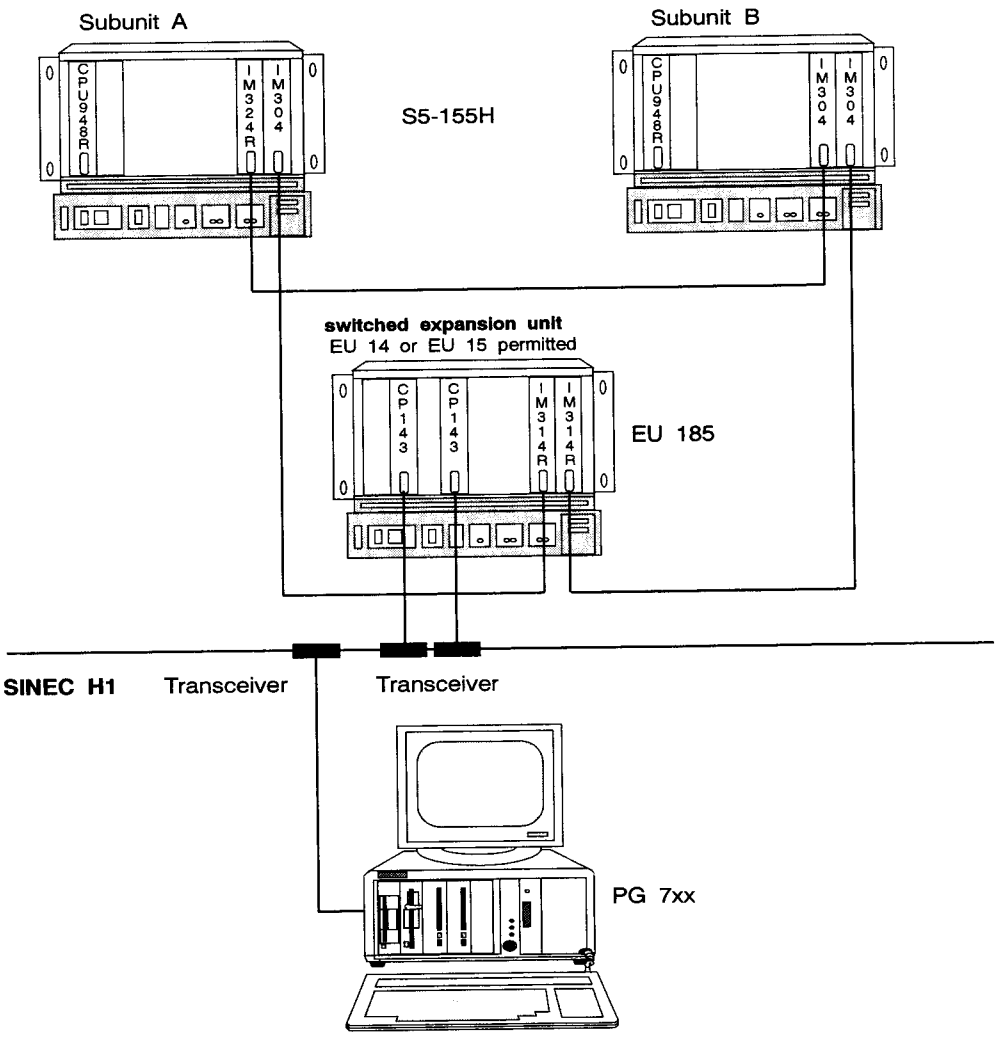


Fig. 10-7 Configuration 1 of the basic configuration: operation with **two** CP 143s (increased availability)

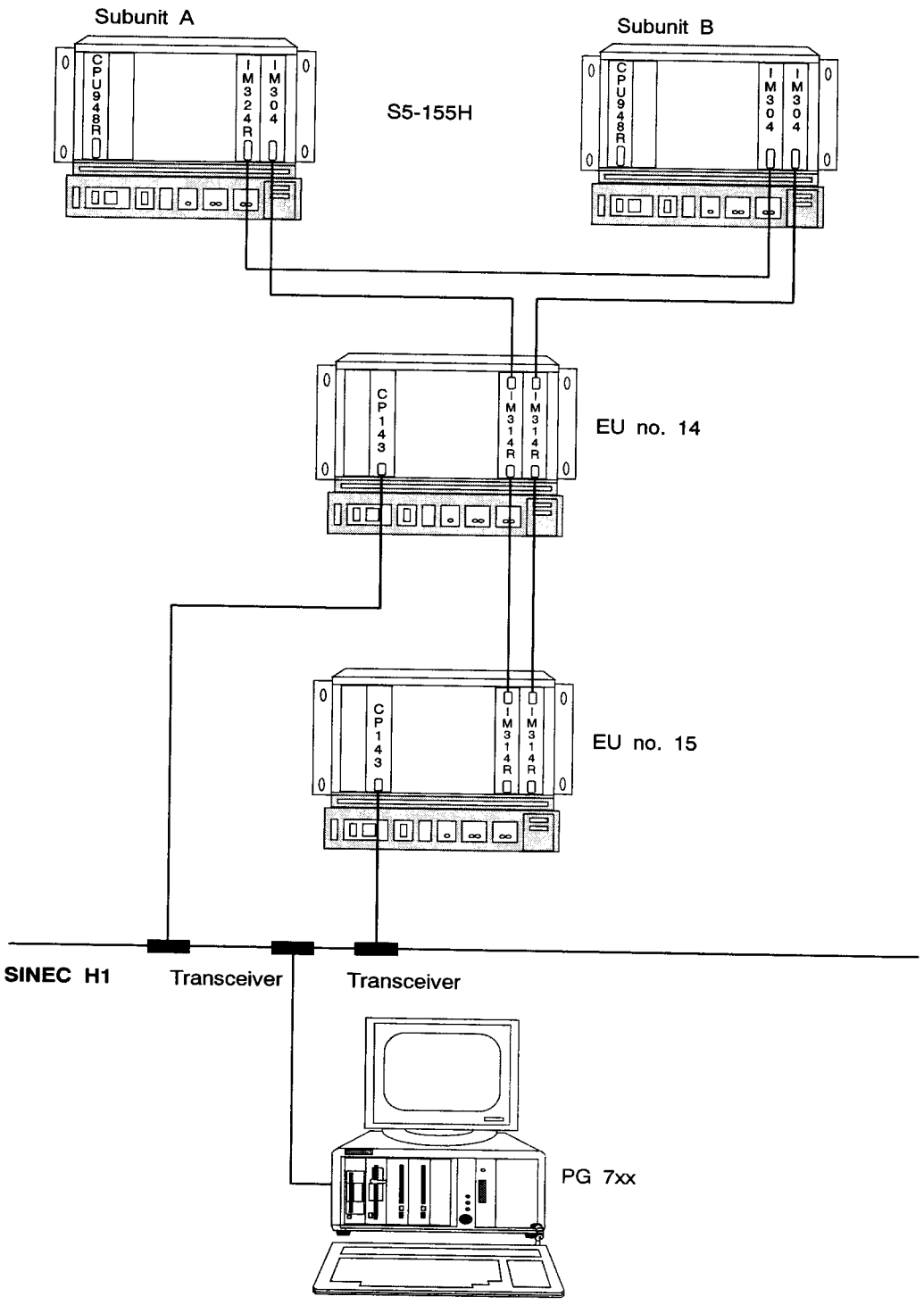


Fig. 10-8 Configuration 2 of the basic configuration: operation with **two** CP 143s (increased availability)

Installation The following steps are necessary during installation:

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Set the interface number (SSNR) on the CP 143 (jumpers):
Select the SSNR according the existing hardware configuration as described below.
Remember the instructions in /6/.
<table>
<tr><th>Possible hardware configuration</th><th>Corresponding SSNR of the CP 143(s)</th></tr>
<tr><td>Basic configuration</td><td>232 or 236</td></tr>
<tr><td>Configuration 1</td><td>232, 236</td></tr>
<tr><td>Configuration 2</td><td>232, 236</td></tr>
</table>
The interface numbers 232 and 236 are reserved exclusively for PG functions via the S5 bus.
For all other CPs and IPs SSNR < 232 must be selected.
Each CP 143 has four pages that are used as follows for communication:
SSNR + 0: for user communication via HDB page
SSNR + 1: for user communication via HDB page
SSNR + 2: system page for PG functions via S5 bus
SSNR + 3: reserved</td></tr>
<tr><td>2</td><td>Plug the CP 143 into the S5-155H (the power to the PLC must be off!).</td></tr>
<tr><td>3</td><td>Connect the PG to the PG interface of the CP 143 and load COM 143 .</td></tr>
<tr><td>4</td><td>Set the same hardware interface number as in step 1 and the Ethernet address in the SYSID of the CP 143 with COM 143.</td></tr>
<tr><td>5</td><td>Load the parameter data on the CP 143:
The parameter data of the CP 143 can be stored
- in an EPROM
or
- in the RAM of the CP 143.

You can transfer the parameter data via the serial interface of the PG 7xx. The individual steps for loading the parameter data of the CP 143 can be found in /6/.</td></tr>
<tr><td>6</td><td>Set the mode selector on the CPU to RUN.</td></tr>
<tr><td>7</td><td>Switch the power for the PLC on again.</td></tr>
<tr><td>8</td><td>Edit the path to the CPU 948R in the STEP 5 bus selection screen.</td></tr>
<tr><td>9</td><td>Select the path via SINEC H1/CP 143 to the CPU 948R in the STEP 5 presets screen.</td></tr>
</table>

Special features

The CPU 948R synchronizes the pages for PG functions via the S5 bus on the CP 143 automatically after POWER UP. The mode selector must be set to RUN.

For cyclic operation, the four interface numbers must be assigned with COM 155H.

In the following situations, the link between the PG and CPU must be re-established:

- After a change from STOP to RUN
  (Owing to this reaction, you **cannot** use the PG function "**status block**" via the S5 bus for **start-up OBs**.
- after a change from RUN to STOP,
- after a master-standby switchover,
- after depassivation
- after a change to the redundant mode.

Note
When you start up with COM 155H the **subunit which has access to the switched expansion unit at the start** is automatically started first (the subunit with access can be recognized by the green LED lit on the assigned IM 314R module).
The other PLC has the loaded user program transferred to it automatically at START-UP. You recognize the master unit by the green LED lit on the assigned IM 314R module.

Specifying and editing bus paths

Specify the paths you want to use in the following test to match your configuration.

Using the STEP 5 software package, edit the bus paths and save them under a suitable name in the path file.
When you load and test the user program, you simply have to enter the file name and path name in the presets screen.
The CPU 948R only accepts one established path for PG communication at any one time.

Paths can only be activated at the PG.

10.5.6 Displaying Problems

The CPU 948R (subunit A or subunit B) sets error bits in its RT area if errors occur during PG functions via the S5 bus.

These bits are set in a parameter assignment error byte (PAFE) for every possible connection and a condition codeword (ANZW) to indicate the current status of the Send and Receive blocks.
The bits are largely the same as those of the handling blocks.

RT area

An information field of 16 words is set up in the RT area of the CPU 948 only when the pages for PG functions exist and the connection to the CP 143 is established. The structure of this area is described below.

> **Note**
> The RT area is **reset** after an OVERALL RESET!
>
> If you use the PG functions via the S5 bus, the RT area is occupied as described below and is then no longer available for other programs (e.g. standard FBs). Please remember this when configuring your system.
>
> As long as there is no connection to the CP 143 (PAFE = 71), for example, because no page with SSNR 232ff or 236ff exists, **no additional information** is entered in the RT area.

Structure of the information field

Address			
E F2E8H	Condition codeword for data direction from CPU to PG for SS 234 or 238		RT 232
E F2E9H	Condition codeword for data direction from CPU to PG for SS 235 or 239		RT 233
E F2EAH	reserved		RT 234
E F2EBH	reserved		RT 235
E F2ECH	Data direction from PG to CPU for SSNR 234 or 238		RT 236
E F2EDH	Data direction from PG to CPU for SSNR 235 or 239		RT 237
E F2EEH	reserved		RT 238
E F2EFH	reserved		RT 239
E F2F0H	reserved	PAFE 234	RT 240
E F2F1H	reserved	reserved	RT 241
E F2F2H	reserved	PAFE 238	RT 242
E F2F3H	reserved		RT 243
E F2F4H			RT 244
E F2F5H			RT 245
E F2F6H			RT 246
E F2F7H			RT 247

Meaning of the PAFE values

All errors occurring in the interaction with the CP 143 are indicated. The following errors can be indicated in PAFE:

PAFE value	Meaning
00H	no error
71H	interface (page) does not exist
81H	interface not ready
91H	interface overloaded
A1H	interface being used by other CPU
B1H	job number of field size illegal (FB SYNCHRON)
C1H	interface not reacting or not in time
D1H	other interface errors, including error messages for the CP

Meaning of value 71H:

The value 71H means that the page does not exist. The PG functions via the S5 bus cannot be started if this message occurs. In this case, check the interface assignment of the CP 143: interface numbers 232 ff or 236 ff must be set (jumpers and in SYSID!).

ANZW

ANZW contains the current status of the Send and Receive blocks. The bits have the following meaning:

High byte	
Bit no.	Meaning
15	not used
14	
13	
12	
11	
10	
9	
8	
Low byte	
7	not used
6	data received
5	data transferred
4	1: error
3	job terminated with error
2	job completed without error
1	0: SEND enabled [1)] 1: SEND disabled
0	0: RECEIVE disabled [1)] 1: RECEIVE enabled

[1)] only for PG functions via the S5 bus

Appendix 11

Contents of Chapter 11

Technical Data of the CPU 9487R and CPU 946R/947R..... 11 - 4

11

Appendix

This chapter contains technical data of the CPU 948R and the CPU 946R/947R as an overview table.

Technical Data of the CPU 948R and CPU 946R/947R

Operation / Processing	CPU 948R	CPU 946R/947R
Typical execution times of bit operations		
with F, I, Q D Formal operand	 0.18 µs 0.70 µs 0.91 µs	 1.4 µs 2.4 µs 2.7 µs
Typical execution times of word operations:		
- Load operations L FY (byte) L FW (word) L FD (double word) - Fixed point arithmetic - Floating point arithmetic	 0.18 µs 0.50 µs 0.71 µs 0.55 ... 3.8 µs 3.3 ... 6.3 µs	 1.5 µs 1.8 µs 2.8 µs 1.0 ... 8.7 µs 12.8 ... 18 µs
Cyclic program execution		
Basic load calling OB 1:	10 ms	30 ms
Additional time for process image update: per DI byte, three-channel: per DI byte, redundant: per DI byte, switched: per DI byte, one-sided: per DQ byte, redundant: per DQ byte, switched: per DQ byte, one-sided:	 200 µs 50 µs 25 µs 40 µs 100 µs 4 µs 25 µs	 400 µs 136 µs 60 µs 86 µs 236 µs 5 µs 64 µs
Additional time for IPC flag transmission: per input: per output:	 25 µs 4 µs	 60 µs 5 µs
Additional time for timer processing depending on timer field length (TFL) TFL =0 TFL #0 n = number of timers running (interval: 10 ms)	 every 10 ms 11.6 µs 11.6 µs +TFL * 0.32 µs	 every 10 ms 4.4 µs 4.4 µs +TFL * 1.3 µs

Operation / Processing	CPU 948R	CPU 946R/947R
Interrupt-driven program execution		
Extension of the cycle time by nesting an empty OB 2 (without STEP 5 operations) at a block boundary	280 µs	430 µs
Reaction time	200 µs	300 µs
Time-controlled program execution		
Extension of the cycle time by nesting an empty OB 13 (without STEP 5 operations) at a command boundary	300 µs	430 µs
Clock pulse for calling the time-controlled program (timed interrupts OB 10 to OB 18)	variable basic clock pulse from 1 to 255 ms. specified relative to 10 ms: 10, 20, 50, 100 200, 500 ms, 1, 2, 5 s or 10, 20, 40, 80, 160, 320, 640 ms, 1.28, 2.56 s	variable basic clock pulse from 1 to 255 ms. specified relative to 10 ms: 10, 20, 50, 100 200, 500 ms, 1, 2, 5 s or 10, 20, 40, 80, 160, 320, 640 ms, 1.28, 2.56 s
Resolution for real time interrupt (OB 9)	every minute, hourly, daily, weekly, monthly, annually, once	–
Resolution for delayed interrupt (OB 6)	1 ms	–
Cycle time monitoring		
Selectable between triggerable	200 ms 1 ... 2550 ms yes	200 ms 1 ... 2550 ms no

11

Operation / Processing	CPU 948R	CPU 946R/947R
Memory capacity		
Size of the user memory module (in Kbytes)	640 or 1664	128 + n * 128 (n = number of submodules in memory module 355 (max. 6 modules in 2 memory modules 355)
Timers, counters, flags		
Number of timers and counters	256 of each	256 of each
Number of flags	2048 flags + 32768 S flags	2048 flags + 32768 S flags

12 Further Reading

Further Reading

/1/ S5-155H
CPU 948R
Pocket Guide

Order no. 6ES5 997-3UR21

/2/ System Manual S5-135U/155U

Order no. 6ES5 998-0SH21

/3/ STEP 5 Manual

Order no. C79000-G8576-C140

/4/ GRAPH 5: Graphic programming of sequential controls under the S5-DOS SIMATIC S5 operating system

Order no. 6ES5 998-1SA01

/5/ Handling Blocks
Standard Function Blocks
CPU 948

Order no. C79000-G8563-C572

/6/ SINEC
Manual
CP 143 with COM 143

Order no. 6GK1970-1AB43-0AB0

/7/ Hans Berger:
Automating with SIMATIC S5-155U

SIEMENS AG ISBN 3-8009-1538-3

/8/ Programmable Controllers
Basic Concepts

SIEMENS AG
Order no. E80850-C293-X-A2

/9/ Catalog ST 59: Programmers
SIMATIC S5

/10/ Catalog ST 54.1: Programmable Controllers
S5-135U, S5-155U and S5-155H

/11/ Catalog ST 57: Standard Function Blocks
and drivers for
programmable controllers of the U series
SIMATIC S5

/12/ Manual SCL

Order no. C79000-G8576-C162

/13/ S5-C-Compiler for S5-155U

Order no. 6ES5 886-1MA21

Indexes

13

Contents of Chapter 13

List of Abbreviations ..... 13 - 3

List of Keywords ..... 13 - 5

List of Tables and Figures ..... 13 - 9

List of Tables ..... 13 - 9
List of Figures ..... 13 - 15

List of Abbreviations

(An explanation of the ISTACK abbreviations can be found in Section 5.4)

ACCU-1 (2, 3, 4)-L	low word in accumulator 1 (2, 3, 4), 16 bit
ACCU-1 (2, 3, 4)-H	high word in accumulator 1 (2, 3, 4), 16 bit
ACCU-1 (2 ,3, 4)-LL	low byte of low word in accumulator 1 (2, 3, 4), 8 bit
ACCU-1 (2, 3, 4)-LH	high byte of low word in accumulator 1 (2, 3, 4), 8 bit
ADF	addressing error
ANZW	condition code word
BASP	disable command output (signal on S5 bus)
BCD	binary coded decimal
BR	base address register
BSTACK	block stack
CC 1, CC 0	condition code bits for digital operations
COR	coordinator module
CP	communications processor
CPU	central processing unit
CSF	control system flowchart
DB	data block
DBA	data block start address (in register 6)
DBL	data block length (in register 8)
DX	extended data block
EPROM	erasable programmable read only memory
ERAB	first scan (bit code)
EU	expansion unit
FB	function block
FX	extended function block
IM	interface module
INT	(system)interrupt
IP	intelligent peripheral module
ISTACK	interrupt stack

KB	call for a non-existent logic block
KDB	opening a non-existent DB/DX-data block
LAD	ladder diagram
LED	light-emitting diode
NAU	power failure
OB	organization block
OR	or (bit code)
OS	overflow latching (word code)
OV	overflow (word code)
PAFE	parameter assignment error byte
PARE	parity error
PB	program block
PEU	power failure on expansion unit
PG	programmer
PI	process image
PII	process image of the inputs
PIQ	process image of the outputs
PLC	programmable controller
QVZ	timeout
RAM	random-access memory
RLO	result of logic operation
SAC	step address counter
SB	sequence block
SPU	operating system processor
STA	status (bit code)
STL	statement list
STS	stop statement
SUF	substitution error
STUEB	BSTACK overflow
STUEU	ISTACK overflow
TRAF	transfer or load error
ZYK	cycle error

List of Keywords

A

Access to DW 255 9-15
ACCU writing conventions 6-5
Accumulators 6-20
Actual operands
  of function blocks 2-29
ADF (addressing error) 5-35
Arithmetic operations 3-59
Assignment list 2-24

B

Basic operations 2-4, 3-19
BASP LED 4-6
BASP signal 4-31
Binary logic operations 3-52
Binary numbers 2-8
Block
  block preheader 2-35
  body 2-13, 2-24, 2-36
  formal operands (block parameters) 2-27
  header 2-13, 2-36
  number 2-12, 2-36, 3-33
  preheader 2-14
Block address list 3-8, 8-13
Block calls 2-16, 3-8, 3-32
Block ID 2-36
Block operations 3-32
Blocks
  deleting 6-14
  generating 6-17
BR register 9-22
BSTACK (block stack) 1-20, 5-5
  evaluating 5-7
  output 5-6

C

CC 1 and CC 0
  see results codes
Checkpoints 10-8
COLD RESTART 4-25
COLD RESTART WITH MEMORY 4-25
Collision of timed interrupts 4-43
Comparison operations 3-32
Control bits 5-5, 5-10
Correcting blocks 2-15
Counter value 3-28
Counters C 1-12
CP 143 10-19
  assigning parameters 10-21
CPU type and ID 8-35
CSF (control system flowchart) 2-4
CYCLE 3-11, 4-33
  interrupt points 4-34
  program execution level 4-31
  user interface OB 1 4-34
Cycle monitoring time 6-47
Cycle time
  current cycle time of S5-155H 8-32
Cyclic communication 4-12
Cyclic processing 1-4
Cyclic program execution 1-15, 4-33
Cyclic timed interrupt 4-37
Cyclic timed interrupts 4-32, 4-41

D

Data block DB 0 2-41, 3-8, 8-13
Data block DB 1 2-41
Data block DB 2 2-41
Data block DX 0 2-41, 7-4
Data block DX 1 2-41
Data blocks
  general 1-12
Data blocks (DB/DX)
  copying from memory card 6-48
  copying/duplicating 6-48
  general 2-13, 2-35
  generating 3-33
  programming 2-37
  structure 2-35
  validity 2-38
Data word 1-12, 2-35, 2-39
Date 6-32
DBA (data block start address) 9-11
DBL (data block length) 9-13
Decimal numbers 2-8
Decrementing 3-68
Defaults, modifying 1-7
Delay interrupts 6-26, 6-29
Delayed interrupt 1-20, 4-32, 4-37 - 4-38, 6-44
Digital logic operations 3-52
Disable interrupts 6-12, 6-23
Dual-port RAM 9-29

E

$\overline{\text{ERAB}}$
  see results codes
Error analysis 5-9
Error DB 5-8

Error in self-test 5-41
Error information 5-5
Errors
avoiding 5-4
cause 5-32
handling 5-29

F

F flags 1-11
FEDBX (error in G DB/GX DX) 5-41
Fixed point numbers 2-9
Flags
multiple use 4-48
Floating point numbers 2-8
Formal operands 2-25, 3-54
Function blocks (FB/FX)
general 2-13, 2-23
programming 2-25
standard function blocks 2-23, 2-33
structure 2-24

G

Global memory
access 9-25
general 9-4
GRAPH 5 2-5

H

HARD STOP 4-4

I

I/Os
modules 1-10
O area 1-10
P area 1-10
Incrementing 3-68
Interface
to system program 1-7, 1-9, 2-18
Interprocessor communication flags
general 3-13
Interrupt-driven processing 1-5
Interrupt-driven program execution 4-36, 4-45
ISTACK (interrupt stack) 4-8
code bits 5-16
contents 5-15
error information 5-5
information in ISTACK 5-16
output 5-6, 5-10

J

Jump operations 3-61

L

LAD (ladder diagram) 2-4
LED display 4-5 - 4-6, 4-15 - 4-16
Library number 2-36
Load operations 3-21, 3-57
Local memory
access 9-24
general 9-4
Logic operations 3-52

M

Mantissa
see floating point number
Memory access 9-6
via the BR register 9-22
via address in ACCU 1 9-8
Memory card 1-19, 3-10
Memory organization 9-4
Mode of operation of CPU 1-4

N

NAU (power failure) 4-23, 4-27
Nesting depth 3-9
Nesting program execution levels 4-9
No operation 3-34

O

O area
see I/Os
Operand areas 1-10
Operating statuses 4-4
OR
see results codes
Organization block (OB)
general 2-12, 2-16
Organization blocks
for communication in SOFT STOP 2-20
Organization blocks (OB)
as user interfaces 2-18

for processing errors 2-20
special function organization blocks 6-4
OS (overflow latching)
see results codes
OV (overflow)
see results codes
OVERALL RESET 4-17

P

P area
see I/Os
Page area/DPR
occupied register 9-30
Pages
accessing 9-29
Parameters for DX 0 1-7
PARE (parity error) 5-37
PG functions 10-4
PG functions via the S5 bus 10-19
error bits 10-31
interface numbers 10-20
technical requirements 10-19
PG screen form
for DX 0 parameter assignment 7-5
PG software 1-18
PLC identification field 8-33
Priority 1-5, 4-8, 4-10, 4-37, 4-46, 7-7
Process image
outputs (PIQ) 1-4
inputs (PII) 1-4
general 1-10
inputs (PII) 1-10
outputs (PIQ) 1-10
updating 4-31
PROCESS INTERRUPT 4-31
Process interrupts
disable 3-74
enable 3-74, 4-47
Process interrupts via input byte IB 0
user interfaces 4-45
Program
storing 3-10
Program blocks (PB) 2-12, 2-16
Program execution level
general 4-9, 6-26
Program execution levels 4-7
Programming
general 1-14
Programming H systems 1-17
Programming language
C with S5-C compiler 1-18
GRAPH 5 1-18
SCL 1-18
STEP 5 1-18
Programming language SCL 1-18
Programming tools 1-18

Q

QVZ (timeout error) 5-34

R

Reaction time 4-48
Reactions with error OBs not loaded 5-30
Real time interrupt 1-20, 4-32, 4-37, 4-39, 6-37
Results codes
ERAB 3-16, 3-20
CC 1 and CC 0 3-18, 3-63
OR 3-17
OS 3-17
OV 3-17
RLO 2-7, 3-17, 3-20
STA 3-17, 3-20
RI/RJ area 8-14
RLO
see results codes
RS/RT area 8-15
assignment of RS area 8-16
RUN
general 4-4, 4-31
RUN LED 4-5

S

S flags 1-11
Self-test 5-43
Sequence blocks (SB) 2-12
Serial link PG - PLC 10-18
Set/reset operation 3-20
Shift operations 3-63
SOFT STOP 4-4
Special functions
calling 6-5
errors in processing 6-6
general 6-4
interfaces to special functions 6-5
new 1-20
STA (status)
see results codes
Standard function blocks
see also function blocks
START-UP 3-11
general 3-11, 4-4, 4-19

13

interruptions 4-27
triggering 4-21, 4-23
STEP 5 operations 3-15
STL (statement list) 2-4
STOP
mode 4-12
STOP LED 4-5
Stop operations 3-34
Structure of the memory area 8-4 - 8-5
Structured programming 2-5
Sub-level 4-10
SUF (substitution error) 5-37
Supplementary operations 2-4
System data 8-15
System data words
bit assignment 8-18
System operations 2-4, 3-61, 9-4
System program 1-6
System program defaults 1-7
System RAM 8-6
System time 6-8, 6-32

T

Time-controlled processing 1-5
Time-controlled program execution 4-37
Timed interrupts 4-32
Timed job 4-39, 6-37
Timer and counter operations 3-26, 3-55
Timers T 1-12
TRAF (load and transfer error) 5-38
Transfer operations 3-21, 3-57
Transferring memory fields 9-19

U

User interfaces
for cyclic program execution 4-34
for delayed interrupt 4-38
for process interrupts 4-45
for real time interrupt 4-39
for start-up 4-26
for timed interrupts 4-42
User memory 1-13, 3-10
User program 1-6, 1-8
processing 3-4, 3-11
see program
storing 1-9
tasks 1-8

W

WEFES/WEFEH (collision of timed interrupts) 5-39

Z

ZYK (cycle time error) 5-36

List of Tables and Figures

List of Tables

Table 2-1 Overview of the organization blocks of the CPU 948R for program execution ..... 2 - 19

Table 2-2 Overview of the organization blocks of the CPU 948R for start-up ..... 2 - 20

Table 2-3 Organization blocks of the CPU 948R for a SOFT STOP ..... 2 - 20

Table 2-4 Overview of the organization blocks of the CPU 948R for error handling ..... 2 - 20

Table 2-5 Overview of the organization blocks of the CPU 948R for special functions ..... 2 - 22

Table 2-5 Permitted formal operands for function blocks ..... 2 - 27

Table 2-6 Permitted actual operands for function blocks ..... 2 - 29

Table 2-7 Data formats permitted in a data block ..... 2 - 37

Table 3-1 Result condition codes of STEP 5 operations ..... 3 - 18

Table 3-2 Binary logic operations ..... 3 - 19

Table 3-3 Set/reset operations ..... 3 - 20

Table 3-4 Load and transfer operations/part 1 ..... 3 - 21

Table 3-5 Load and transfer operations/part 2 ..... 3 - 22

Table 3-6 Timer and counter operations ..... 3 - 26

Table 3-7 Arithmetic operations ..... 3 - 31

Table 3-8 Comparison operations ..... 3 - 32

Table 3-9 Block operations ..... 3 - 32

Table 3-10 NOP/display/stop operations ..... 3 - 34

Table 3-11 Binary logic operations with formal operands ..... 3 - 52

Table 3-12 Digital logic operations . . . . . 3 - 52

Table 3-13 Bit test operations . . . . . 3 - 53

Table 3-14 Set/reset operations with formal operands . . . . . 3 - 54

Table 3-15 Set and reset operations. . . . . 3 - 54

Table 3-16 Timer and counter operations with formal operands . . . . . 3 - 55

Table 3-17 Load and transfer operations with formal operands . . . . . 3 - 57

Table 3-18 Load and transfer operations with special operands. . . . . 3 - 58

Table 3-19 Arithmetic operation ENT . . . . . 3 - 59

Table 3-20 Supplementary arithmetic operations . . . . . 3 - 60

Table 3-21 Jump operations . . . . . 3 - 61

Table 3-22 Shift operations . . . . . 3 - 63

Table 3-23 Conversion operations . . . . . 3 - 65

Table 3-24 Decrement/increment operation . . . . . 3 - 68

Table 3-25 Processing operations . . . . . 3 - 68

Table 3-26 I/O operations . . . . . 3 - 74

Table 3-27 Other operations . . . . . 3 - 74

Table 3-28 Meaning of the abbreviations in UAMW . . . . . 3 - 75

Table 4-1 Meaning of the LEDs "RUN" , "STOP" and "SYSFAULT" . . . . . 4 - 5

Table 4-2 Program execution levels . . . . . 4 - 8

Table 4-3 Differences between COLD RESTART and COLD RESTART WITH MEMORY . . . 4 - 25

Table 4-4 Sets of intervals and intervals of the TIMED INTERRUPTS . . . . . 4 - 41

Table 4-5 Timed interrupt collision IDs: meaning of the bits in ACCU-1-L. . . . . 4 - 43

Table 4-6 User interfaces for process interrupts . . . . . 4 - 45

Table 5-1 Meaning of the control bits SYSTEM DESCRIPTION ..... 5 - 12

Table 5-2 Meaning of the control bits STOP CAUSE ..... 5 - 12

Table 5-3 Meaning of the control bits START-UP IDS ..... 5 - 13

Table 5-4 Meaning of the control bits ERROR IDS ..... 5 - 14

Table 5-5 Meaning of the ISTACK IDs for errors ..... 5 - 16

Table 5-6 ISTACK IDs CAUSE OF INTERRUPTION ..... 5 - 18

Table 5-7 Group error bits in the status word of the error DB ..... 5 - 23

Table 5-8 Status byte in the H flag word ..... 5 - 27

Table 5-9 Control byte in the H flag word ..... 5 - 28

Table 5-10 The organization blocks called in case of errors ..... 5 - 29

Table 6-1 Overview of the special functions available with the CPU 948R ..... 6 - 4

Table 6-2 Error IDs of OB 121 in ACCU-1-L ..... 6 - 9

Table 6-3 Error IDs of OB 122 in ACCU-1-L ..... 6 - 13

Table 6-4 Results bits of OB 124 ..... 6 - 15

Table 6-5 Result IDs of OB 124 in ACCU-1-LL ..... 6 - 16

Table 6-6 Result bits of OB 125 ..... 6 - 18

Table 6-7 Result IDs of OB 125 in ACCU-1-LL ..... 6 - 19

Table 6-8 Error IDs of OB 141 in ACCU-1-L ..... 6 - 25

Table 6-9 Error IDs of OB 142 in ACCU-1-L ..... 6 - 28

Table 6-10 Error IDs of OB 143 in ACCU-1-L ..... 6 - 31

Table 6-11 Error IDs of OB 150 in ACCU-1-L ..... 6 - 35

Table 6-12 Error IDs of OB 151 in ACCU-1-L ..... 6 - 40

Table 6-13 Assignment of "timed job - time parameters" ..... 6 - 41

Table 6-14 Error IDs of OB 153 ..... 6 - 45

Table 6-15 Result codes for OB 254/255 ..... 6 - 50

Table 6-16 Result IDs for OB 254/255 in ACCU-1-L ..... 6 - 51

Table 8-1 Structure of the memory area ..... 8 - 4

Table 8-2 Assignment of the RS area ..... 8 - 16

Table 8-3 Bits in RS 0 (image of IB 0) ..... 8 - 18

Table 8-4 Bits of RS 1 (current process interrupts) ..... 8 - 19

Table 8-5 Bits of RS 5 (current cycle time of the S5-155H) ..... 8 - 20

Table 8-6 Bits of RS 7 (PLC stop IDs) ..... 8 - 21

Table 8-7 Bits of RS 8 (start and start-up IDs) ..... 8 - 22

Table 8-8 Bits of RS 16 (error area output bytes 0 to 15) ..... 8 - 23

Table 8-9 Bits of RS 9 (QVZ test) ..... 8 - 24

Table 8-10 RS 75: general errors ..... 8 - 27

Table 8-11 RS 75: Errors in DX 0 or DB 1 ..... 8 - 28

Table 8-12 RS 76 to RS 78: Parameter types ..... 8 - 29

Table 8-13 Bits of BS 251 (current cycle time of the S5-155H) ..... 8 - 32

Table 9-1 Operations for indirect memory access using registers ..... 9 - 8

Table 9-2 16-bit register for LIR/TIR ..... 9 - 9

Table 9-3 32-bit register for LDI/TDI ..... 9 - 17

Table 9-4 Operations for field transfer ..... 9 - 19

Table 9-5 Memory areas permitted for TNW, TXB and TXW ..... 9 - 19

Table 9-6 Load and arithmetic operations with the BR register ..... 9 - 22

Table 9-7 Register-register operations ..... 9 - 23

Table 9-8 Operations for accessing the local memory ..... 9 - 24

Table 9-9 Operations for access to the global memory organized in bytes . . . . . 9 - 27

Table 9-10 Operations for access to the global memory organized in words. . . . . 9 - 28

Table 9-11 Operations for access to pages organized in bytes . . . . . 9 - 31

Table 9-12 Operations for access to pages organized as words . . . . . 9 - 33

Table 10-1 Functions for installation and testing . . . . . 10 - 4

List of Figures

Fig. 1-1 Tasks of the system program ..... 1 - 6

Fig. 1-2 Structure of a STEP 5 user program ..... 1 - 9

Fig. 2-1 Methods of representation in the STEP 5 programming language ..... 2 - 5

Fig. 2-2 Example of block storage in the user memory ..... 2 - 15

Fig. 2-3 Block calls that enable processing of a program block ..... 2 - 17

Fig. 2-4 Structure of a function block (FB/FX) ..... 2 - 24

Fig. 2-5 Range of validity of an opened data block ..... 2 - 40

Fig. 3-1 Principle of cyclic program execution ..... 3 - 4

Fig. 3-2 Example of the organization of the user program according to the program structure ... 3 - 6

Fig. 3-3 Example of the organization of the user program according to the structure of the controlled system ..... 3 - 7

Fig. 3-4 Nested logic block calls ..... 3 - 8

Fig. 3-5 Example of block nesting depth ..... 3 - 9

Fig. 3-6 Load and transfer operations in a byte-oriented memory area ..... 3 - 23

Fig. 3-7 Load and transfer operations in a word-oriented memory area ..... 3 - 24

Fig. 4-1 Front panel of the CPU 948R with display and operating elements ..... 4 - 4

Fig. 4-2 Program execution levels ..... 4 - 7

Fig. 4-3 Principle of changing level and the ISTACK ..... 4 - 11

Fig. 4-4 Program execution after POWER UP ..... 4 - 14

Fig. 4-5 Program execution after a cycle interruption ..... 4 - 14

Fig. 4-6 Cyclic program execution with the S5-155H ..... 4 - 33

Fig. 5-1 Example of the first screen form page "OUTPUT ISTACK": control bits ..... 5 - 11

Fig. 5-2 Example of a screen page "OUTPUT ISTACK" ..... 5 - 15

Fig. 5-3 Example of evaluating the ISTACK. ..... 5 - 20

Fig. 5-4 Structure of the error DB ..... 5 - 22

Fig. 5-5 Structure of an error field ..... 5 - 25

Fig. 6-1 Effect of the "roll-up" function ..... 6 - 22

Fig. 6-2 Effect of the "roll-down" function ..... 6 - 22

Fig. 7-1 PG screen form for assigning parameters to DX 0 / Part 1 ..... 7 - 5

Fig. 7-2 PG screen form for assigning parameters to DX 0 / Part 2 ..... 7 - 6

Fig. 8-1 Memory assignment in CPU 948R/overview ..... 8 - 5

Fig. 8-2 Memory assignment for the system RAM/Part 1 ..... 8 - 6

Fig. 8-3 Memory assignment of the system RAM/Part 2 ..... 8 - 7

Fig. 8-4 Address areas for the I/O (8 bits) on the S5 bus ..... 8 - 8

Fig. 8-5 Example: Location of blocks in memory ..... 8 - 11

Fig. 8-6 Block addresses in DB 0 ..... 8 - 13

Fig. 9-1 Global and local memory ..... 9 - 5

Fig. 9-2 Access to local or global areas using absolute addresses ..... 9 - 7

Fig. 9-3 LIR/TIR with 8-bit memory areas (byte-oriented) ..... 9 - 10

Fig. 9-4 LIR/TIR with 16-bit memory areas (word-oriented) ..... 9 - 10

Fig. 9-5 Using the DBA register ..... 9 - 12

Fig. 9-6 Using the DBL register ..... 9 - 14

Fig. 9-7 Transferring memory fields ..... 9 - 21

Fig. 9-8 Transfer operations from one register to another ..... 9 - 24

Fig. 10-1 PG display of the memory configuration with older PG software ..... 10 - 9

Fig. 10-2 Interface assignment of the PG functions via the S5 bus ..... 10 - 20

Fig. 10-3 One-sided operation, configuration 1: with **one** one-sided CP 143 ..... 10 - 22

Fig. 10-4 One-sided operation, configuration 2: with **two** one-sided CP 143s ..... 10 - 22

Fig. 10-5 Paths between PG and CPU 948R and assignment of the pages of the CP 143 in one-sided operation ..... 10 - 25

Fig. 10-6 Basic configuration with switched EU: operation with **one** CP 143 ..... 10 - 26

Fig. 10-7 Configuration 1 of the basic configuration: operation with **two** CP 143s (increased availability) ..... 10 - 27

Fig. 10-8 Configuration 2 of the basic configuration: operation with **two** CP 143s (increased availability) ..... 10 - 28

Siemens AG
AUT 1282
Östliche Rheinbrückenstr. 50
D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: .....
Your Title: .....
Company Name: .....
Street: .....
City, Zip Code: .....
Phone: .....

Please check any industry that applies to you:

- ❑ Automotive
- ❑ Chemical
- ❑ Electrical Machinery
- ❑ Food
- ❑ Instrument and Control
- ❑ Nonelectrical Machinery
- ❑ Petrochemical
- ❑ Pharmaceutical
- ❑ Plastic
- ❑ Pulp and Paper
- ❑ Textiles
- ❑ Transportation
- ❑ Other



Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1.Do the contents meet your requirements? ☐

2.Is the information you need easy to find? ☐

3.Is the text easy to understand? ☐

4.Does the level of technical detail meet your requirements? ☐

5.Please rate the quality of the graphics/tables: ☐

Additional comments:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Siemens AG
AUT 1282
Östliche Rheinbrückenstr. 50
D-76181 Karlsruhe
Federal Republic of Germany

From:

Your Name: .....
Your Title: .....
Company Name: .....
Street: .....
City, Zip Code: .....
Phone: .....

Please check any industry that applies to you:

- ❑ Automotive
- ❑ Chemical
- ❑ Electrical Machinery
- ❑ Food
- ❑ Instrument and Control
- ❑ Nonelectrical Machinery
- ❑ Petrochemical
- ❑ Pharmaceutical
- ❑ Plastic
- ❑ Pulp and Paper
- ❑ Textiles
- ❑ Transportation
- ❑ Other



Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please do not forget to state the title, order number and release of your manual.

Title of Your Manual: .....

Order No. of Your Manual: ..... Release: .....

Please give each of the following questions your own personal mark within the range from 1 (very good) to 5 (poor).

1.Do the contents meet your requirements? ☐

2.Is the information you need easy to find? ☐

3.Is the text easy to understand? ☐

4.Does the level of technical detail meet your requirements? ☐

5.Please rate the quality of the graphics/tables: ☐

Additional comments:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....