

# **SIEMENS**

## **SIMATIC S5**

### **S5-155U**

### **Programmable Controller (CPU 946/947)**

**Manual**

**Order No. 6ES5998-0UM22  
Release 07**

## Contents

|  |                      |           |
|--|----------------------|-----------|
| Warnings<br>Information<br>Suggestions/Corrections             | C79000-R8576-C647    |           |
| Product Summary<br>CPU 946/947                                 | C79000-T8576-C647-01 | <b>1</b>  |
| Installation Guide<br>Programmable Controllers of the U Series | C79000-B8576-C452-04 | <b>2</b>  |
| S5-155U/S5-155H Central Controller<br>Instructions             | C79000-B8576-C380-05 | <b>3</b>  |
| CPU 946/947<br>Instructions                                    | C79000-B8576-C532-06 | <b>4</b>  |
| 355 Memory Module<br>Instructions                              | C79000-B8576-C382-06 | <b>5</b>  |
| 923 C Coordinator<br>Instructions                              | C79000-B8576-C349-06 | <b>6</b>  |
| Multiprocessor Operation<br>Instructions                       | C79000-B8576-C500-02 | <b>7</b>  |
| Multiprocessor Communication<br>User's Guide                   | C79000-B8576-C468-05 | <b>8</b>  |
| CPU 946/947<br>Programming Guide                               | C79000-B8576-C637-03 | <b>9</b>  |
| <b>Space</b> for Pocket Guide<br>CPU 946/947, CPU 946R/947R    | C79000-B8576-C054-01 | <b>10</b> |
| Appendix   | C79000-A8576-C647-01 | <b>11</b> |
|  |                      | <b>12</b> |

The Pocket Guide CPU 946/947, CPU 946R/947R (C79000-B8576-C111-02) is included in the front pocket of this Manual.

## Summary of Product Number and Corresponding Documentation for the S5-155U Programmable Controller

The following table shows the assignment of the product documentation to the individual products. The listed sections of the documentation relate to the manual S5-155U (CPU 946/947) with the order no. 6ES5 998-0UM22, release 07.

| Product  |                               | Product Documentation   |
|--|-------------------------------|---|
| Order No.  | Title                         |   |
| 6ES5 155-3UA11<br>6ES5 155-3UA21   | Central Controller<br>S5-155U | <p><b>Instructions:</b><br/>(Part 3): Central Controller S5-155U/155H<br/>C79000-B8576-C380-05</p> <p>(Part 7): Multiprocessor Operation in the<br/>Central Controllers S5-135U/155U<br/>C79000-B8576-C500-02</p> <p><b>User's Guide</b><br/>(Part 8): Multiprocessor Communication S5-135U<br/>Programmable Controller, CPU 922 and 928<br/>S5-155U Programmable Controller, CPU 946/947<br/>C79000-B8576-C468-05</p> <p><b>Installation Guide</b><br/>(Part 2): Programmable Controllers of the<br/>U Series<br/>C79000-B8576-C452-04</p> |
| 6ES5 946-3UA11<br>6ES5 946-3UA12<br>6ES5 946-3UA21<br>6ES5 947-3UA11<br>6ES5 947-3UA21 | CPU 946/947                   | <p><b>Instructions</b><br/>(Part 4): CPU 946/947<br/>C79000-B8576-C532-06</p> <p><b>Programming Guide</b><br/>(Part 9): S5-155U Programmable Controller, CPU<br/>946/947 and S5-155H Programmable Controller,<br/>CPU 946R/947R<br/>C79000-B8576-C637-03</p> <p><b>List of Operations</b> (in the front pocket)<br/>CPU 946/947, CPU 946R/947R<br/>C79000-B8576-C111-02</p>   |
| 6ES5 355-3UA11   | 355 Memory Module             | <p><b>Instructions</b><br/>(Part 5): 355 Memory Module<br/>C79000-B8576-C382-06</p>   |
| 6ES5 923-3UC11   | 923 C Coordinator             | <p><b>Instructions</b><br/>(Part 6): 923 C Coordinator<br/>C79000-B8576-C349-06</p>   |

## **Preface**

This manual provides an overview of the structure and functions of the programmable logical controller S5-155U. It explains how you configure, program, test and start your programmable controller.

This manual is intended as a guide for you to learn to use the PLC and to make optimum use of the features of the device.

This manual is intended for engineers, programmers and maintenance personnel who have a general knowledge of programmable controller concepts.

If you have any questions which have not been answered in this manual, please contact your local Siemens representative.





## **How To Use This Manual**

The following information is intended to make it easier for you to use the S5-155U manual, (order no. 6ES5 998-0UM22).

### **Overview of Contents**

- **Part 1 : Product Summary**

This part provides an overview of the structure of the S5-155U programmable controller with CPU 946R/947R. This overview provides you with the basic information to aid your understanding of the following chapters.

- **Part 2 : Programmable Controllers of the U Series - Installation Guide**

This part provides the information about how the programmable controllers of the U series must be structured. Different configurations, power supply, wiring, fans, temperature monitoring, safety measures and interference suppression are described.

- **Part 3 : 155U Central Controller - Instructions**

This part provides the instructions to the S5-155U central controller. Both the hardware as well as the installation procedure, start-up procedure and the maintenance of the central controller are described. After installation is complete, you will find the information in this section that you need to guarantee trouble free operation of the central controller.

- **Part 4 : CPU 946/947 - Instructions**

This part provides the instructions with a technical description and information about the installation procedure and use of the CPU 946/947; the standard central unit of the SIMATIC S5-155U programmable controller.

- **Part 5 : 355 Memory Module - Instructions**

This part provides the instructions on the memory module. The instructions describe which modules you can use and what you must note about the installation and operation of the individual modules.

- **Part 6 : 923C Coordinator - Instructions**

Here you will find a description of how the modules function and what you must know about operating the coordinator.

- **Part 9 : STEP 5 Programming Instructions for the S5-155U**

The mode of operation and application of the CPU 946/947 are described in this section. Comprehensive information is available so that you can make optimum use of the S5-155U programmable controller.

- **Part 11 : Appendix**

The order numbers of all the components and spare parts mentioned in this manual are listed here.

## **Index**

Each part of this manual has its own index.

## **Training**

For information about training courses in connection with this device, please contact your local Siemens representative.

## Warning

### **Risks involved in the use of so-called SIMATIC-compatible modules of non-Siemens manufacture**

"The manufacturer of a product (SIMATIC in this case) is under the general obligation to give warning of possible risks attached to his product. This obligation has been extended in recent court rulings to include parts supplied by other vendors. Accordingly, the manufacturer is obliged to observe and recognize such hazards as may arise when a product is combined with products of other manufacture.

**For this reason, we feel obliged to warn our customers who use SIMATIC products not to install so-called SIMATIC-compatible modules of other manufacture in the form of replacement or add-on modules in SIMATIC systems.**

Our products undergo a strict quality assurance procedure. We have no knowledge as to whether outside manufacturers of so-called SIMATIC-compatible modules have any quality assurance at all or one that is nearly equivalent to ours. These so-called SIMATIC-compatible modules are not marketed in agreement with Siemens; we have never recommended the use of so-called SIMATIC-compatible modules of other manufacture. The advertising of these other manufacturers for so-called SIMATIC-compatible modules wrongly creates the impression that the subject advertised in periodicals, catalogues or at exhibitions had been agreed with us. Where so-called SIMATIC-compatible modules of non-Siemens manufacture are combined with our SIMATIC automation systems, we have a case of our product being used contrary to recommendations. Because of the variety of applications of our SIMATIC automation systems and the large number of these products marketed worldwide, we cannot give a concrete description specifically analyzing the hazards created by these so-called SIMATIC-compatible modules. It is beyond the manufacturer's capabilities to have all these so-called SIMATIC-compatible modules checked for their effect on our SIMATIC products. If the use of so-called SIMATIC-compatible modules leads to defects in a SIMATIC automation system, no warranty for such systems will be given by Siemens.

In the event of product liability damages due to the use of so-called SIMATIC-compatible modules, Siemens are not liable since we took timely action in warning users of the potential hazards involved in so-called SIMATIC-compatible modules."

# Safety-Related Guidelines for the User

## 1 General

This manual provides the information required for the intended use of the particular product. The documentation is written for technically qualified personnel such as engineers, programmers or maintenance specialists who have been specially trained and who have the specialized knowledge required in the field of instrumentation and control.

A knowledge of the safety instructions and warnings contained in this manual and their appropriate application are prerequisites for safe installation and commissioning as well as safety in operation and maintenance of the product described. Only qualified personnel as defined in section 2 have the specialized knowledge that is necessary to correctly interpret the general guidelines relating to the safety instructions and warnings and implement them in each particular case.

This manual is an inherent part of the scope of supply even if, for logistic reasons, it has to be ordered separately. For the sake of clarity, not all details of all versions of the product are described in the documentation, nor can it cover all conceivable cases regarding installation, operation and maintenance. Should you require further information or face special problems that have not been dealt with in sufficient detail in this documentation, please contact your local Siemens office.

We would also point out that the contents of this product documentation shall not become a part of or modify any prior or existing agreement, commitment or legal relationship. The Purchase Agreement contains the complete and exclusive obligations of Siemens. Any statements contained in this documentation do not create new warranties or restrict the existing warranty.

## 2 Qualified Personnel

Persons who are **not qualified** should not be allowed to handle the equipment/system. Non-compliance with the warnings contained in this manual or appearing on the equipment itself can result in severe personal injury or damage to property. Only **qualified personnel** should be allowed to work on this equipment/system.

Qualified persons as referred to in the safety guidelines in this manual as well as on the product itself are defined as follows:

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

## 3 Danger Notices

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

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

### Danger

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

### Warning

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

### Caution

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

### Note

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

### Important

If in this manual "Important" should appear in bold type, drawing attention to any particularly information, the definition corresponds to that of "Warning", "Caution" or "Note".

## 4 Proper Usage

- The equipment/system or the system components may only be used for the applications described in the catalog or the technical description, and only in combination with the equipment, components and devices of other manufacturers as far as this is recommended or permitted by Siemens.
- The product described has been developed, manufactured, tested and the documentation compiled in keeping with the relevant safety standards. Consequently, if the described handling instructions and safety guidelines described for planning, installation, proper operation and maintenance are adhered to, the product, under normal conditions, will not be a source of danger to property or life.



### Warning

- After opening the housing or the protective cover or after opening the system cabinet, certain parts of this equipment/system will be accessible, which could have a dangerously high voltage level.
- Only suitably qualified personnel should be allowed access to this equipment/system.
- These persons must be fully conversant with any potential sources of danger and maintenance measures as set out in this manual.
- It is assumed that this product be transported, stored and installed as intended, and maintained and operated with care to ensure that the product functions correctly and safely.

## 5 Guidelines for the Planning and Installation of the Product

The product generally forms a part of larger systems or plants. These guidelines are intended to help integrate the product into its environment without it constituting a source of danger.

The following facts require particular attention:



### Note

Even when a high degree of safety has been designed into an item of automation equipment by means of multichannel configuration, it is still imperative that the instructions contained in this manual be exactly adhered to. Incorrect handling can render ineffective the preventive measures incorporated into the system to protect it against dangerous faults, and even create new sources of danger.

The following advice regarding installation and commissioning of the product should - in specific cases - also be noted.



### Warning

- Follow strictly the safety and accident prevention rules that apply in each particular case.
- Units which are designed as built-in units may only be operated as such, and table-mounted or portable equipment only with its casing closed.
- In the case of equipment with a permanent power connection which is not provided with an isolating switch and/or fuses which disconnect all poles, a suitable isolating switch or fuses must be provided in the building wiring system (distribution board). Furthermore, the equipment must be connected to a protective ground (PE) conductor.
- For equipment or systems with a fixed connecting cable but no isolating switch which disconnects all poles, the power socket with the grounding pin must be installed close to the unit and must be easily accessible.
- Before switching on the equipment, make sure that the voltage range setting on the equipment corresponds to the local power system voltage.
- In the case of equipment operating on 24 V DC, make sure that proper electrical isolation is provided between the mains supply and the 24 V supply. Only use power supply units to IEC 364-4-41 or HD 384.04.41 (VDE 0100 Part 410).
- Fluctuations or deviations of the power supply voltage from the rated value should not exceed the tolerances specified in the technical specifications. Otherwise, functional failures or dangerous conditions can occur in the electronic modules/equipment.
- Suitable measures must be taken to make sure that programs that are interrupted by a voltage dip or power supply failure resume proper operation when the power supply is restored. Care must be taken to ensure that dangerous operating conditions do not occur even momentarily. If necessary, the equipment must be forced into the "emergency off" state.
- Emergency tripping devices in accordance with EN 60204/IEC 204 (VDE 0113) must be effective in all operating modes of the automation equipment. Resetting the emergency off device must not result in any uncontrolled or undefined restart of the equipment.



### Caution

- Install the power supply and signal cables in such a manner as to prevent inductive and capacitive interference voltages from affecting the automation functions.
- Automation equipment and its operating elements must be installed in such a manner as to prevent unintentional operation.
- Automation equipment can assume an undefined state in the case of a wire break in the signal lines. To prevent this, suitable hardware and software measures must be taken when interfacing the inputs and outputs of the automation equipment.

### 6 Active and Passive Faults in Automation Equipment

- Depending on the particular task for which the electronic automation equipment is used, both **active** as well as **passive** faults can result in a **dangerous** situation. For example, in drive control, an active fault is generally dangerous because it can result in an unauthorized startup of the drive. On the other hand, a passive fault in a signalling function can result in a dangerous operating state not being reported to the operator.
- This differentiation of the possible faults and their classification into dangerous and non-dangerous faults, depending on the particular task, is important for all safety considerations in respect of the product supplied.



#### Warning

---

In all cases where a fault in an automation equipment can result in severe personal injury or substantial damage to property, ie. where a dangerous fault can occur, additional external measures must be taken or equipment provided to ensure or force safe operating conditions even in the event of a fault (e.g. by means of independent limit monitors, mechanical interlocks etc.).

### 7 Procedures for Maintenance and Repair

If measurement or testing work is to be carried out on an active unit, the rules and regulations contained in the "VBG 4.0 Accident prevention regulations" of the German employers liability assurance association (Berufsgenossenschaften) must be observed. Particular attention is drawn to paragraph 8 "Permissible exceptions when working on live parts". Use only suitable electrical tools.



#### Warning

---

- Repairs to an item of automation equipment may only be carried out by **Siemens service personnel** or an **authorized Siemens repair center**. For replacement purposes, use only parts or components that are contained in the spare parts list or listed in the "Spare parts" section of this manual. Unauthorized opening of equipment and improper repairs can result in loss of life or severe personal injury as well as substantial property damage
- Before opening the equipment, always remove the power plug or open the disconnecting switch.
- Only use the fuse types specified in the technical specifications or the maintenance instructions of this manual.
- Do not throw batteries into an open fire and do not carry out any soldering work on batteries (danger of explosion). Maximum ambient temperature 100°C. Lithium batteries or batteries containing mercury should not be opened or recharged. Make sure that the same type is used when replacing batteries.
- Batteries and accumulators must be disposed of as classified waste.
- The following points require attention when using monitors:  
Improper handling, especially the readjustment of the high voltage or fitting of another tube type can result in excessive X-ray radiation from the unit. The license to operate such a modified unit automatically lapses and the unit must not be operated at all.

The information in this manual is checked regularly for updating and correctness and may be modified without prior notice. The information contained in this manual is protected by copyright. Photocopying and translation into other languages is not permitted without express permission from Siemens.



## Guidelines for Handling Electrostatically Sensitive Devices (ESD)

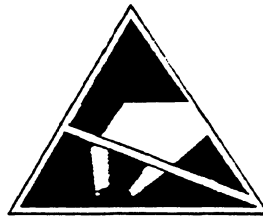
### 1 What is ESD?

VLSI chips (MOS technology) are used in practically all SIMATIC S5 and TELEPERM M modules. These VLSI components are, by their nature, very sensitive to overvoltages and thus to electrostatic discharge:

They are therefore defined as  
"Electrostatically Sensitive Devices"

"ESD" is the abbreviation used internationally.

The following warning label on the cabinets, subracks and packing indicates that electrostatically sensitive components have been used and that the modules concerned are susceptible to touch:



**ESDs** can be destroyed by voltage and energy levels which are far below the level perceptible to human beings. Such voltages already occur when a component or a module is touched by a person who has not been electrostatically discharged. Components which have been subjected to such overvoltages cannot, in most cases, be immediately detected as faulty; the fault occurs only after a long period in operation.

An electrostatic discharge

- of 3500 V can be felt
- of 4500 V can be heard
- must take place at a minimum of 5000 V to be seen.

**But** just a fraction of this voltage can already damage or destroy an electronic component.

The typical data of a component can suffer due to damage, overstraining or weakening caused by electrostatic discharge; this can result in temporary fault behavior, e.g. in the case of

- temperature variations,
- mechanical shocks,
- vibrations,
- change of load.

Only the consequent use of protective equipment and careful observance of the precautions for handling such components can effectively prevent functional disturbances and failures of ESD modules.

### 2 When is a Static Charge Formed?

One can never be sure whether the human body or the material and tools which one is using are not electrostatically charged.

Small charges of 100 V are very common; these can, however, very quickly rise up to 35 000 V.

Examples of static charge:

- |                                      |                |
|--------------------------------------|----------------|
| - Walking on a carpet                | up to 35 000 V |
| - Walking on a PVC flooring          | up to 12 000 V |
| - Sitting on a cushioned chair       | up to 18 000 V |
| - Plastic desoldering unit           | up to 8 000 V  |
| - Plastic coffee cup                 | up to 5 000 V  |
| - Plastic bags                       | up to 5 000 V  |
| - Books, etc. with a plastic binding | up to 8 000 V  |

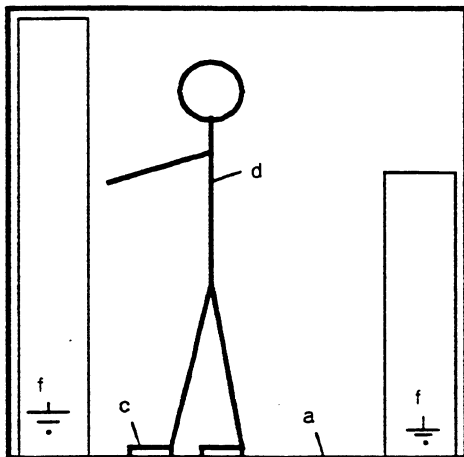
### 3 Important Protective Measures against Static Charge

- Most plastic materials are highly susceptible to static charge and must therefore be kept as far away as possible from ESDs.
- Personnel who handle ESDs, the work table and the packing must all be carefully grounded.

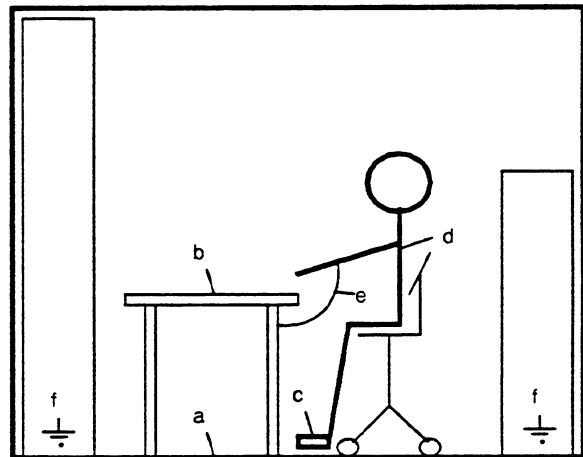
### 4 Handling of ESD Modules

- One basic rule to be observed is that electronic modules should be touched by hand only if this is necessary for any work required to be done on them. Do not touch the component pins or the conductors.
- Touch components only if
  - the person is grounded at all times by means of a wrist strap
  - or
  - the person is wearing special anti-static shoes or shoes with a grounding strip.
- Before touching an electronic module, the person concerned must ensure that (s)he is not carrying any static charge. The simplest way is to touch a conductive, grounded item of equipment (e.g. a blank metallic cabinet part, water pipe, etc.) before touching the module.
- Modules should not be brought into contact with insulating materials or materials which take up a static charge, e.g. plastic foil, insulating table tops, synthetic clothing, etc.
- Modules should only be placed on conductive surfaces (table with anti-static table top, conductive foam material, anti-static plastic bag, anti-static transport container).
- Modules should not be placed in the vicinity of monitors, TV sets (minimum distance from screen > 10 cm).

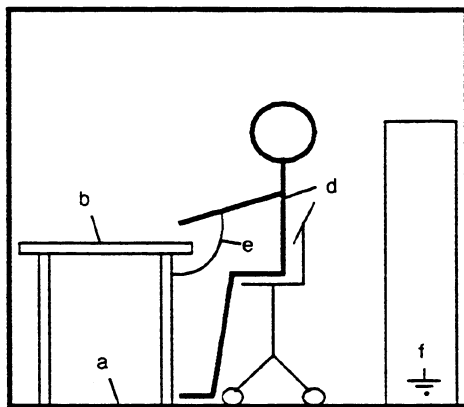
The diagram below shows the required protective measures against electrostatic discharge.



Standing position



Standing/sitting position



Sitting position

- a Conductive flooring
- b Anti-static table
- c Anti-static shoes
- d Anti-static coat
- e Grounding wrist strap
- f Grounding connection of the cabinets

## 5 Measurements and Modification to ESD Modules

- Measurements on modules may only be carried out under the following conditions:
  - The measuring equipment is grounded (e.g. via the PE conductor of the power supply system) or
  - when electrically isolated measuring equipment is used, the probe must be discharged (e.g. by touching the metallic casing of the equipment) before beginning measurements.
- Only grounded soldering irons may be used.

## 6 Shipping of ESD Modules

Anti-static packing material must always be used for modules and components, e.g. metalized plastic boxes, metal boxes, etc. for storing and dispatch of modules and components.

If the container itself is not conductive, the modules must be wrapped in a conductive material such as conductive foam, anti-static plastic bag, aluminium foil or paper. Normal plastic bags or foils should not be used under any circumstances.

For modules with built-in batteries ensure that the conductive packing does not touch or short-circuit the battery connections; if necessary cover the connections with insulating tape or material.

# SIEMENS

## SIMATIC S5

---

CPU 946/947 Product Summary

C79000-T8576-C647-01

---

## CPU 946/947 Product Summary

The product summary provides an overview of the functions and characteristics of the CPU 946/947. It briefly describes general performance features and fields of application; the programming languages used are also mentioned.

### Performance features

In the S5-155U programmable controller the CPU 946/947 can be used in single and in multiprocessor mode. It is designed for fast processing of all STEP 5 operations. With a maximum of 896 kbytes this CPU offers a large memory capacity.

The most outstanding features are:

- Max. 896 kbyte memory for data and user programs, as RAM and/or EPROM. 128 kbyte RAMs are already integrated in the CPU.
- - 2048 flags
  - 32768 S flags
  - 256 timers
  - 256 counters
- Processing from up to
 

|                                     |                       |                        |
|-------------------------------------|-----------------------|------------------------|
| 4096                                | binary inputs/outputs | each                   |
| 192                                 | analog inputs/outputs | each                   |
| Via special interface modules up to | 520 000               | digital inputs/outputs |
|                                     | 32 000                | analog inputs/outputs  |

is possible.

### Programming language and program processing

Use the STEP 5 programming language /1/ to program the CPU 946/947. The types of representation for STEP 5 are the ladder diagram (LAD), control system flow chart (CSF) and statement list (STL). Furthermore GRAPH 5 is available which is used to program sequential controls (step sequences).

The CPU 946/947 allows user programs which are

- cyclic
- alarm-controlled
- time-controlled (max. 9 clock grids).

A hardware clock is also integrated.

The average processing time of 1K instructions is

- 1.4 ms            for binary instructions
- 1.7 ms            for command mix (60 % bits, 40 % words)

In the Programming Instructions in Part 9 of this manual you will find detailed information on programming the CPU 946/947. All operations and their running times are described in the Pocket Guide in Part 10 of this manual.

# SIEMENS

## SIMATIC S5

Programmable Controllers of the U Series

---

Installation Guide

C79000-B8576-C452-04

---

|            | Page   |
|------------|--|
| <b>1</b>   | <b>Introduction to Use of Installation Guidelines ..... 3</b>  |
| <b>2</b>   | <b>Fundamentals of EMC ..... 4</b>   |
| <b>3</b>   | <b>Selection and Design of Cabinets ..... 6</b>  |
| <b>3.1</b> | <b>Selection Criteria..... 6</b>   |
| <b>3.2</b> | <b>Types of Cabinets ..... 6</b>   |
| <b>3.3</b> | <b>Specifications When Designing a Cabinet ..... 8</b>   |
| <b>3.4</b> | <b>Power Loss ..... 10</b>   |
| 3.4.1      | Power Loss in Cabinet and Cabinet Cooling ..... 10   |
| 3.4.2      | Example of Calculating Cabinet Type ..... 11   |
| <b>3.5</b> | <b>Example of a Cabinet Design..... 12</b>   |
| <b>4</b>   | <b>Design and Connection of Power Supplies ..... 15</b>  |
| <b>4.1</b> | <b>Internal Power Supply for Central Controllers and Expansion Units ..... 15</b>  |
| <b>4.2</b> | <b>Load Power Supply ..... 16</b>  |
| <b>4.3</b> | <b>Electrical Design with Process Peripherals ..... 17</b>   |
| 4.3.1      | Power Supply for CCs, EUs and Process Peripherals from Grounded Battery<br>or Grounded Power Supply Units ..... 18                     |
| 4.3.2      | Power Supply for CCs, EUs and Process Peripherals from Centrally<br>Grounded Battery or Centrally Grounded Power Supply Units ..... 19 |
| 4.3.3      | Power Supply for CCs, EUs and Process Peripherals from Non-Grounded<br>Battery or Non-Grounded Power Supply Units ..... 20             |
| <b>4.4</b> | <b>Load Power Supply from Two Power Supply Units..... 21</b>   |
| 4.4.1      | Non-floating Modules ..... 21  |
| 4.4.2      | Floating Modules..... 22   |
| <b>5</b>   | <b>Wiring Layout ..... 23</b>  |
| <b>5.1</b> | <b>Wiring Layout Inside a Cabinet..... 23</b>  |
| <b>5.2</b> | <b>Wiring Layout Outside Cabinets..... 24</b>  |
| <b>5.3</b> | <b>Wiring Layout Outside Buildings ..... 25</b>  |
| <b>5.4</b> | <b>Equipotential Bonding..... 25</b>   |



|            |  |           |
|------------|--|-----------|
| <b>6</b>   | <b>Cabinet Wiring and Design with Respect to EMC.....</b>          | <b>26</b> |
| <b>6.1</b> | <b>Grounding of Inactive Metal Components.....</b>                 | <b>27</b> |
| <b>6.2</b> | <b>Shielding of Devices and Cables.....</b>                        | <b>28</b> |
| <b>6.3</b> | <b>Use of Special Noise Suppression Measures .....</b>             | <b>30</b> |
| <b>6.4</b> | <b>Example of an EMC-compatible Cabinet Design .....</b>           | <b>32</b> |
| <b>6.5</b> | <b>Checklist for EMC-compatible Cabinet Design.....</b>            | <b>34</b> |
| <b>7</b>   | <b>Framework and Wall Mounting of SIMATIC S5 Controllers .....</b> | <b>36</b> |
| <b>8</b>   | <b>Lightning Protection Measures .....</b>                         | <b>38</b> |
| <b>9</b>   | <b>Safety Measures.....</b>  | <b>39</b> |
| <b>9.1</b> | <b>Protection against Indirect Contact .....</b>                   | <b>39</b> |

# 1 Introduction to Use of Installation Guidelines

This document is intended for planning, installation and commissioning engineers.

The installation guidelines are divided into the following sections:

- **Chapter 1** Introduction to Use of Installation Guidelines
- **Chapter 2** Fundamentals of EMC  
This section provides a summary of the rules you must observe to ensure electromagnetic compatibility.
- **Chapter 3** Selection and Design of Cabinets  
This section lists criteria which must be considered when selecting the cabinet. The conditions resulting from the power loss of the modules used and the ambient temperature are considered in particular. The power losses of SIMATIC modules are listed.
- **Chapter 4** Design and Connection of Power Supplies  
This section provides information you must observe for the electrical connection of the power supply to CCs, EUs and process peripherals.
- **Chapter 5** Wiring Layout  
This section describes how you can achieve a high interference-resistance of your programmable controller by using a correct wiring layout.
- **Chapter 6** Cabinet Wiring and Design with Respect to EMC  
This section describes the measures required to ensure EMC of your programmable controller. It shows how you can prevent fundamental errors when designing and wiring cabinets. A checklist is provided to check the EMC-compatible cabinet design.
- **Chapter 7** Framework and Wall Mounting  
This section describes what you must observe if you fit your SIMATIC controller in a framework or on a wall.
- **Chapter 8** Lightning Protection Measures  
This section provides information about the measures you should take to protect outdoor cables and lines for SIMATIC devices from lightning strikes.
- **Chapter 9** Safety Measures  
This section provides a summary of the measures you must always take when planning the use of programmable controllers in order to prevent danger during operation. The regulations CENELEC HD 384.4.41 (IEC 364-4-41) (VDE 0100) and EN 60 204 (IEC 204-1) (VDE 0113) must be applied in order to carry out these measures.

We recommend that users who are using a SIMATIC S5 controller for the first time follow the installation guidelines right from the beginning when planning the control system.

We strongly recommend that all users particularly observe the sections and paragraphs concerned with preventing danger (especially Chapter 9) and protection from sources of error (especially Chapter 6). Even if you are an experienced user, check your design using the checklist in Chapter 6.

## 2 Fundamentals of EMC

### Definition of EMC

**Electromagnetic compatibility (EMC) means that an electrical device is able to function correctly in a defined electromagnetic environment without disturbing other devices in its vicinity.**

It is frequently sufficient to observe a few elementary rules to achieve electromagnetic compatibility (EMC). It is essential for you to observe the following four rules when installing your programmable controller.

#### **Rule 1: Make sure there is a perfectly functioning reference ground (central grounding point)**

- Connect the central controller and expansion units to the central grounding point in a star-shaped configuration without loops.
- Protect the PLC from external influences by installing it in a cabinet or housing. Incorporate the cabinet or housing into the ground system.
- Shield electromagnetic fields resulting from inductors (transformers, motors, contactor coils) from the PLC using barriers (steel, highly permeable material).
- Use metal plug housings (not plastic) for screened data transmission lines.

#### **Rule 2: Use a large-area ground.**

- Connect all inactive metal components with a large-area contact and a low impedance.
- Establish a central connection between the inactive metal components and the central grounding point.
- The screw connections on inactive, painted metal components should be made using NOMEL contact washers <sup>1)</sup>.
- Do not forget to incorporate the screen bar into the ground system. This means that the screen bar itself must be connected to ground via a large-area contact.
- Aluminium components are unsuitable for grounding.

<sup>1)</sup> Contact washer Siemens standard 70093 available from  
- Siemens ANL A443 Werkzeug 8520 Erlangen  
- Teckentrup GmbH und Co. KG, Postfach 120, D-5974 Herscheid 2,  
- NOMEL S.A. Tour Franklin, Cedex 11, F-92081 Paris.  
- or from your local Siemens representative

**Rule 3: Plan the wiring layout and ensure that the plan is kept to**

- Divide the cables into groups and route them separately.  
(power cables, power supply cables, signal lines, data lines)
- Always route power cables and signal cables in separate ducts or bundles.
- All the cables should only be fed into the cabinet from one side.
- Route the signal cables as close as possible to grounded components (e.g. cabinet members).
- We recommend the twisting of the forward and return lines of individually routed cables.

**Rule 4: Ensure that your cables are well shielded**

- Data transmission cables should be screened and connected at both ends.
- Analog cables should be screened and the screen connected at one or both ends.
- The cable screens must be connected at the cabinet inlet to the screen bar using a large-area contact and secured with clamps.
- Route the screen up to the module without interruptions.

## **3 Selection and Design of Cabinets**

### **3.1 Selection Criteria**

The following criteria must be observed when selecting and dimensioning a cabinet:

- (i) Ambient conditions
- (ii) Quantity and type of power supplies and subracks to be used
- (iii) Total power loss of components present in the cabinet.

The ambient conditions present where the cabinet is located (temperature, humidity, dust, chemical influences) define the required degree of protection of the cabinet (IP XX) as shown in Fig. 1. Further information on degrees of protection can be found in IEC 529 and DIN 40050.

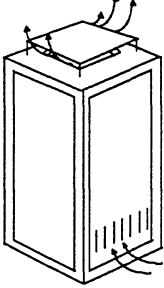
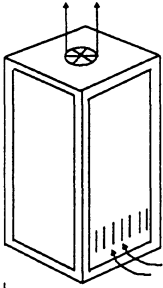
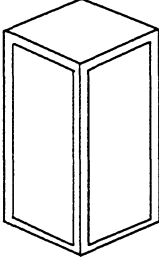
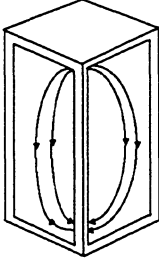
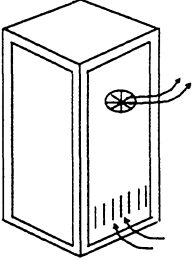
The required design of the cabinet is described in Section 3.3. Make sure that the maximum ambient temperature for the modules is not exceeded.

This involves both the ambient temperature outside the cabinet and the power loss in the cabinet.

It may be necessary to provide a fan or heat exchanger if the power loss is too high. A typical cabinet design is shown using an example at the end of the section.

### **3.2 Types of Cabinets**

The following diagram shows a summary of the most common types of cabinet. It also shows the principle of heat dissipation used, the maximum achievable heat dissipation and the degree of protection.

| Open cabinets   |   | Closed cabinets  |  |  |
|---|---|--|--|--|
| Through-ventilation by natural convection   | Increased through-ventilation using external fan (with filter)                    | Natural convection   | Forced circulation using fan assembly, improvement of natural convection   | Forced circulation using heat exchanger <sup>2)</sup> , external ventilation inside and outside  |
|              |  |   |    |   |
| Heat dissipation primarily by natural thermal convection, small portion via the cabinet wall. | Increased heat dissipation through increased air movement.                        | Heat dissipation only through the cabinet wall; only low power loss permissible. Heat accumulation usually occurs in the top of the cabinet. | Heat dissipation only through the cabinet wall. Forced ventilation of the internal air results in improved heat dissipation and prevention of heat accumulation. | Heat dissipation through exchange between heated internal air and cool outside air. The increased surface area of the heat exchanger and the forced circulation between the inside and outside air permit good heat dissipation. |

Temperature difference between ambient temperature and cabinet temperature (measured at top in the cabinet): 20 °C<sup>4)</sup>

Power loss P<sup>3)</sup> with cabinet dimensions of 2200 mm x 600 mm x 600 mm

Installation as single unit:

|  |   |  |  |  |
|--|---|--|--|--|
| Up to approx. 700 W                      | Up to approx. 2700 W (approx. 1400 W with very fine filter) | Up to approx. 260 W                      | Up to approx. 360 W                      | Up to approx. 1700 W                     |
| Degree of protection IP 20 <sup>1)</sup> | Degree of protection IP 20 <sup>1)</sup>                    | Degree of protection IP 54 <sup>1)</sup> | Degree of protection IP 54 <sup>1)</sup> | Degree of protection IP 54 <sup>1)</sup> |

Fig. 1 Types of cabinets

- <sup>1)</sup> The location and the ambient conditions present there are decisive for selection of the type of cabinet protection (see IEC 529 and DIN 40050).
- <sup>2)</sup> See Catalog NV21 for heat exchangers.
- <sup>3)</sup> The values only apply if the guidelines for installation are adhered to (for further details refer to the following section).
- <sup>4)</sup> If other temperature differences are present, refer to the temperature characteristics of the cabinet manufacturer.

### 3.3 Specifications When Designing a Cabinet

You must first define the components to be fitted in the cabinet. Then calculate the total power loss of the individual components. The following specifications must be observed:

- The expansion units can be accommodated together with the respective central controller in one cabinet, or also in several cabinets (centralized or distributed). See Section 2 for the installation dimensions of the subracks.
- As a result of the required spacing between devices and the maximum permissible installation height for control elements, a maximum of three U-type devices can be arranged one above the other (see Fig. 2).

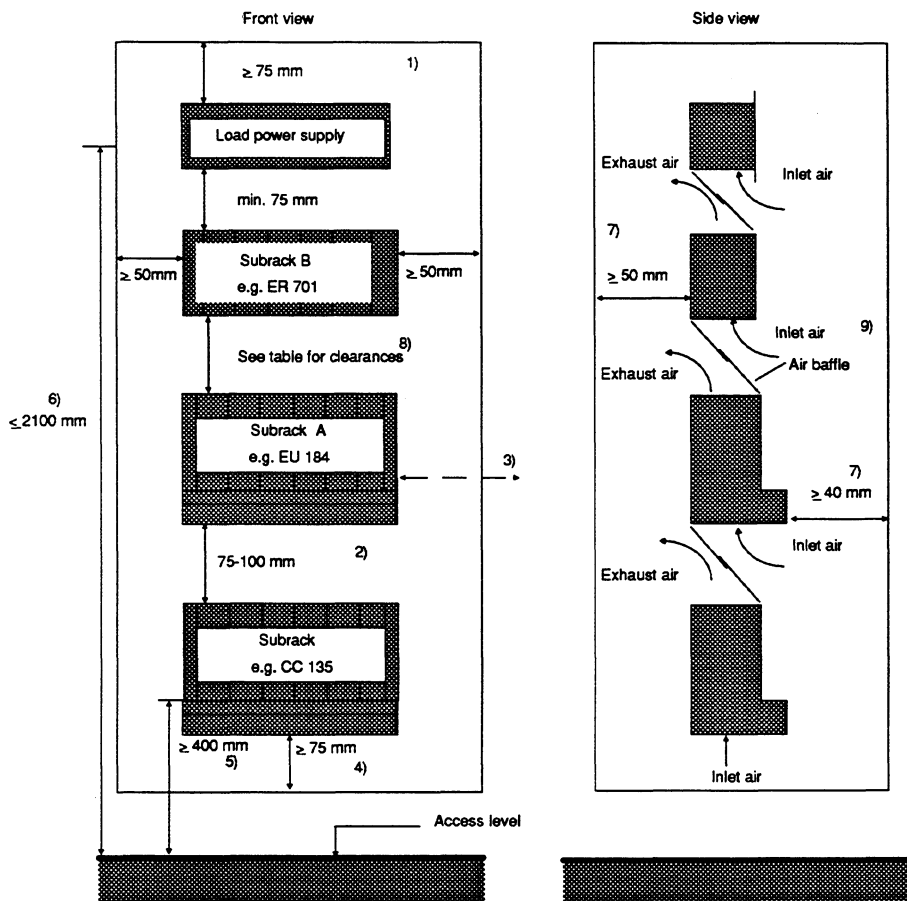



Fig. 2 Installation dimensions for SIMATIC controllers in cabinet

- 1) Min. 75 mm with closed cabinet roof. Smaller distances are possible with a perforated cabinet roof and an additional, separate ventilation roof.
- 2) Min. 75 mm space for inlet air and exhaust air, max. 100 mm because of cable length between the CC/EU interface modules.
- 3) Max. spacing of 400 mm possible (min. 50 mm) when connecting devices next to one another (with IM 312).
- 4) Min. 75 mm from obstructions (large equipment) in the inlet air area.
- 5) Min. installation height above access level 400 mm for control elements, 200 mm for connections.
- 6) Max. installation height for control elements: 2100 mm to VDE 0106, Part 100, 2000 mm to EN 60 204 (IEC 204-1) (VDE 0113).
- 7) Space for air circulation (400 mm deep cabinets are sufficient).
- 8) See Table 2-1 for the distances between subracks A and B.
- 9) The installation of air baffles is recommended to provide a better air supply.

**Note:**

 The expansion unit with the largest power loss should be positioned as the top unit.


- If subracks are combined (CC and EU), the clearances listed in Table 1 must be observed.

| Subrack A <sup>1)</sup>                        | Subrack B <sup>1)</sup> | Minimum clearance | Maximum clearance   |
|--|-------------------------|-------------------|---|
| S5-135U/155U<br>or<br>S5-115U<br>or<br>S5-100U | S5-135U                 | 75 mm             | Limited by the length of the connection cables to the interface modules |
|  | S5-115U with fan        | 60 mm             |   |
|  | S5-115U without fan     | 100 mm            |   |
|  | S5-100U                 | 75 mm             |   |

Table 1 Required clearance between subracks

<sup>1)</sup> See Fig. 2, Installation dimensions for SIMATIC controllers in cabinet

**Note:**

 If subracks from the S5-135U/155U series are used together with subracks of the S5-115U in the same cabinet, ensure that the rear panels of the subracks have the same clearance to the rear panel of the cabinet. This results in improved air circulation.

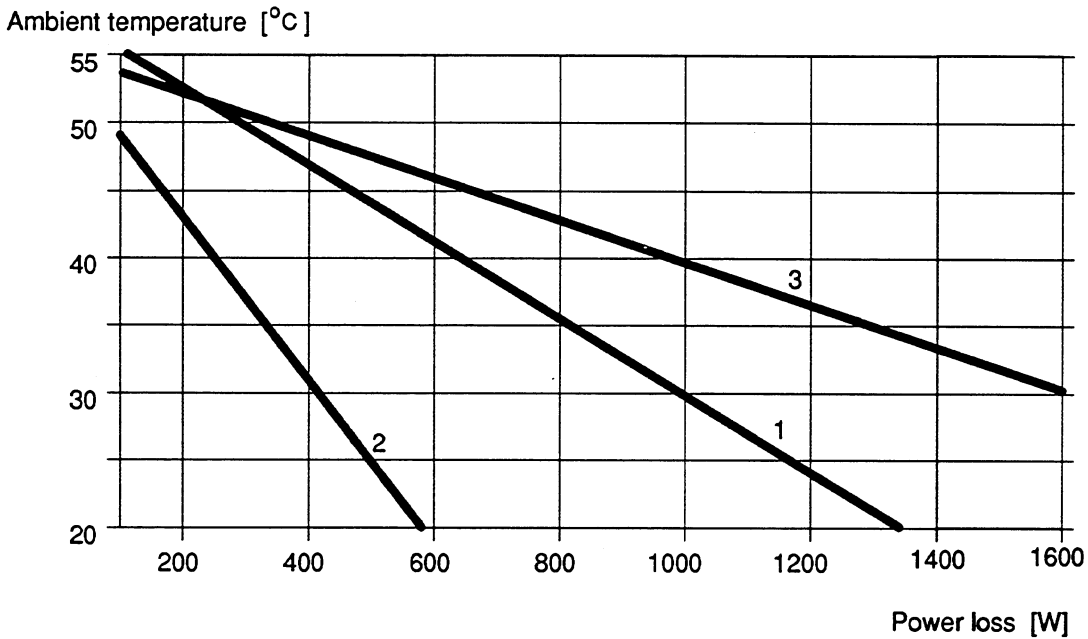


### 3.4 Power Loss

#### 3.4.1 Power Loss in Cabinet and Cabinet Cooling

The power loss that can be dissipated from a cabinet depends on the cabinet design, its ambient temperature and the arrangement of units in the cabinet.

Fig. 3 shows the permissible ambient temperature of a cabinet with dimensions of 600 mm x 600 mm x 2200 mm depending on the power loss. The values indicated only apply to the arrangement of units in the cabinet as shown in Fig. 2. You can obtain more information from Catalogs NV21 and ET1.



Curve 1 Cabinet (open) with through-ventilation by natural convection.  
 Curve 2 Closed cabinet with natural convection and internal forced circulation using fan.  
 Curve 3 Closed cabinet with heat exchanger. Heat exchanger size 11/6 (920 mm x 460 mm x 111 mm).

Fig. 3 Maximum cabinet environment temperature depending on the power loss

**Note:**

➔ When fitting the subracks of the S5-135U/155U series, the maximum power loss which can be dissipated by the fans must not be exceeded. The max. dissipated power loss per unit with an inlet temperature of 55 °C is 250 W. This value is increased by 20 W for each reduction in the inlet temperature by 1 °C.

**Caution:**

⚠ Modules with a hard disk drive can only be used up to an ambient temperature of 50 °C.

### 3.4.2 Example of Calculating Cabinet Type

The following example shows the maximum permissible ambient temperature for different types of cabinet with the same power loss.

Example:

The following configuration is present:

|  |       |
|--|-------|
| 1 central controller                                       | 200 W |
| 2 expansion units, each with 250 W power loss              | 500 W |
| 1 load power supply, 24 V/40 A, 6EV1 362-5BK00 (full load) | 200 W |
| Total power loss:  | 900 W |

Fig. 3 shows the max. ambient temperatures for a total power loss of 900 W:

| Cabinet design (see Fig. 1)                            | Max. ambient temperature |
|--|--------------------------|
| Closed, with natural convection and forced circulation | (use not possible)       |
| Open with through-ventilation                          | Approx. 33 °C            |
| Closed, with heat exchanger                            | Approx. 42 °C            |
| Framework/wall   | Max. 55 °C               |

The power losses of the modules can be found in the technical data in the catalogs or in the manuals.

If these values are not listed in the technical data, they can be calculated easily from the power consumption. To do this, multiply the value of the power consumption by the appropriate voltage.

#### Examples:

- CPU 928B: power consumption 4 A/5 V → power loss = 20 W
- CP 143: power consumption 4 A/5 V  
0.5 A/15 V  
0.04 A/24 V → power loss approx. 21 W
- IM 304 power consumption 1.5 A/5 V → power consumption = 7.5 W

### **3.5 Example of a Cabinet Design**

Figs. 4 and 5 show a design using the example of a metric 8MF cabinet (2200 mm x 600 mm x 600 mm). This design has a number of advantages:

- Universal application
- Independent of the cabinet width (550 to 1200 mm possible)
- The units can be installed asymmetrically; you thus gain more space on one side for routing signal cables.
- All devices can be installed and removed from the front, even after initial installation. The M6 screws must be premounted on the 19-inch cabinet member at the correct mounting height. You can then hook in the subrack and tighten the screws (one-man installation)
- The separate cable routing for analog, digital and power supply lines in cable ducts increases the resistance to mutual interferences between the signals.

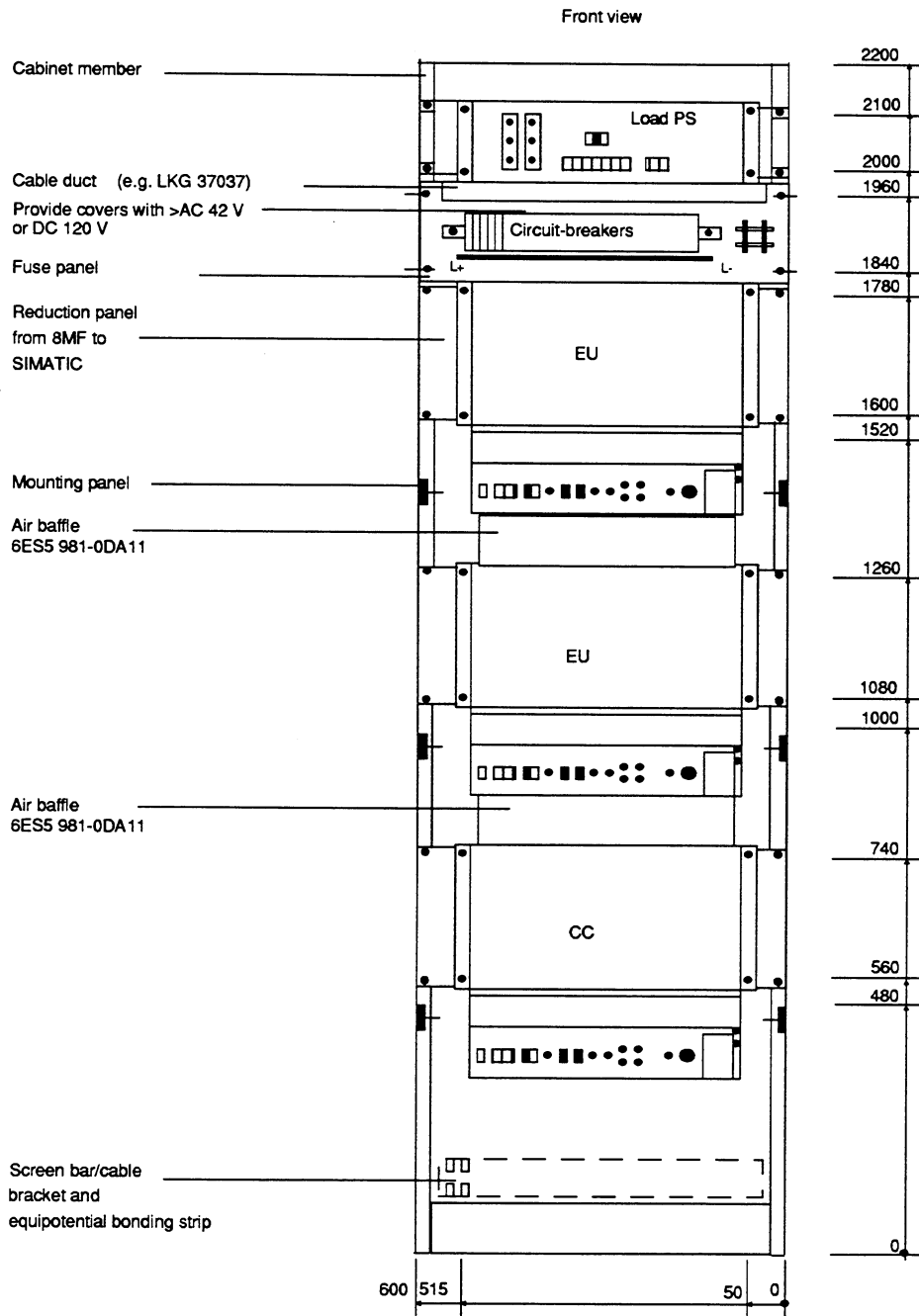


Fig. 4 Front view of 8MF cabinet

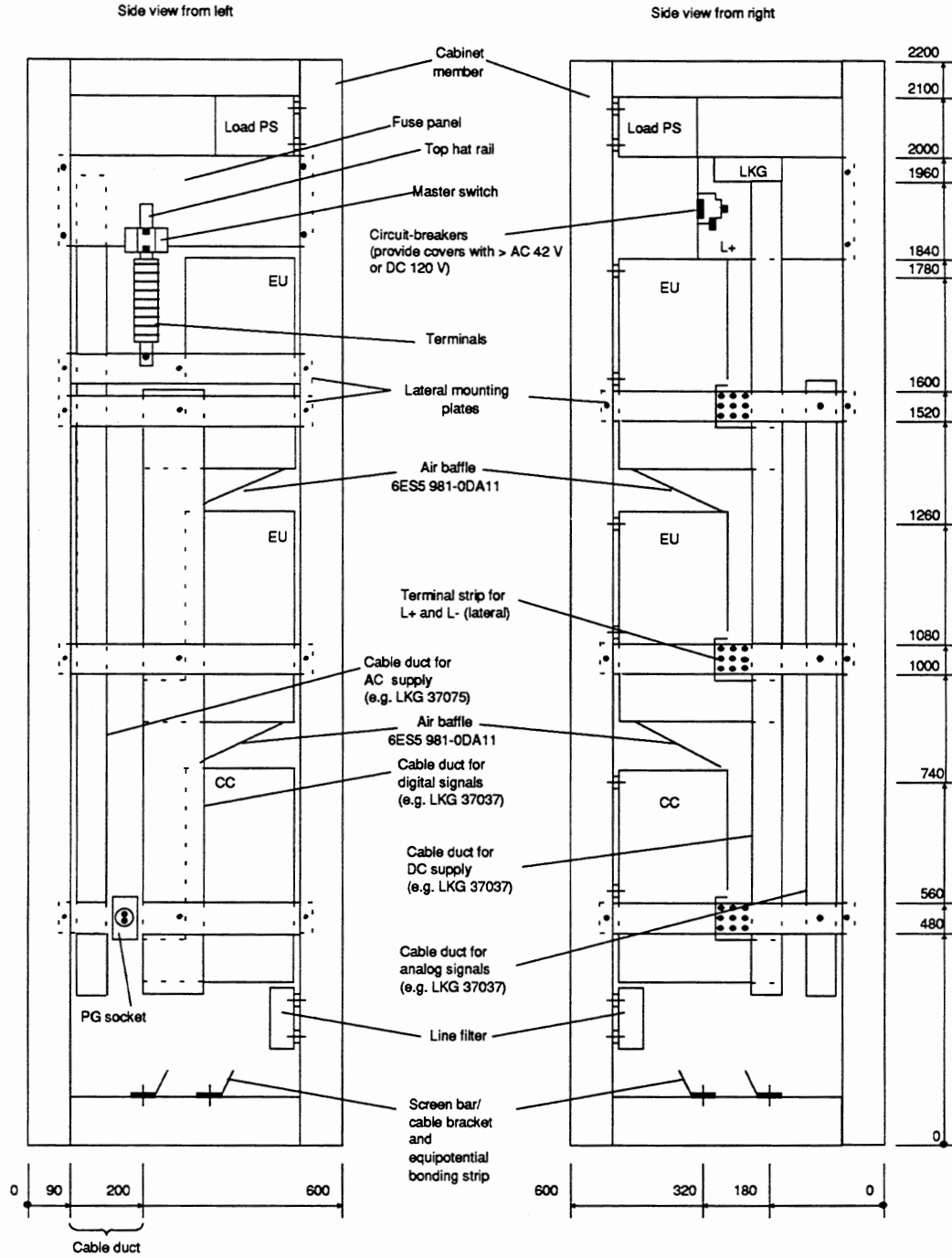


Fig. 5 Side views of 8MF cabinet

## **4 Design and Connection of Power Supplies**

The following section provides information you must observe for the electrical connection of the power supply.

Different power supplies are required for SIMATIC S5 systems:

- Internal power supply for central controllers and expansion units

The internal power supply of the SIMATIC modules is obtained from power supply units in the form of plug-ins. These power supply units are fixed components of the CC and the EU. You can find their technical data in the Instructions of the respective units and in the catalogs.

- Load power supply for the I/O modules as well as sensors and actuators.

### **4.1 Internal Power Supply for Central Controllers and Expansion Units**

The power supplies fitted in the CCs and EUs deliver the internal DC voltages of 5 V, 15 V and 24 V from the input voltage of 120/230 V AC or 24 V DC.

When equipping the CCs and EUs, ensure that the rated current of the respective power supply is not exceeded. You can find the current consumption of the individual modules with the 5 V supply e.g. in the catalogs and the Instructions of the respective module (Technical Data).

Floating and non-floating power supplies are available for the input voltage of 24 V DC.

The permissible input voltage for power supplies with a rated input voltage of 24 V DC is:

- Static DC 20 to 30 V.

The permissible input voltage is as follows for power supplies with a rated input voltage of 230/120 V AC:

- With rated voltage 230 V: 187 to 253 V AC
- With rated voltage 120 V: 93 to 127 V AC.

## 4.2 Load Power Supply

The series 6EV 13.. power supply units from Siemens (output currents 20 and 40 A) can be used to supply the I/O modules as well as the CC and EU power supplies with the input voltage of 24 V DC. Detailed information can be found in Catalog ET1.

The following must be observed when dimensioning load power supplies for digital output modules (S5-135U/155U series):

- To protect the cables and lines from overcurrents and to protect the modules from short-circuits, additional fuses are present on the modules in addition to the electronic short-circuit protection (in the power supply). The fuses also serve as protection if the power supply connections are reversed.
- The electronic short-circuit protection for digital outputs only responds when 2-3 times the rated current is exceeded. You should therefore make sure that the load power supply can supply the current required to trigger the short-circuit protection of an output.
- Note when selecting the load power supply, and taking into consideration all connected output loads, that two to three times the rated output current can flow briefly at the output in the event of a short-circuit before the pulsed electronic short-circuit protection takes effect. This excess current is generally present with non-regulated load power supply units.
- In the case of regulated load power supply units, especially with small output currents up to 20 A, the rated output current of the load power supply must be dimensioned such that several times the rated current can flow in the event of a short-circuit.



**Caution:**

Safe electrical isolation according to CENELEC HD 384.4.41 (IEC 364-4-41) Part 4 (VDE 0100) or VDE 0160 must be guaranteed with all power supply units used for SIMATIC S5 devices and modules. All electrically-isolated Siemens power supplies of the 6EV13... series satisfy this condition.

### **4.3 Electrical Design with Process Peripherals**

The following section shows various designs of power supplies for CCs, EUs and process peripherals.

The following are possible:

- Grounded power supply
- Centrally grounded power supply
- Non-grounded power supply.

You must observe the following fundamental points when designing the electrical configuration of the process peripherals:

- A master switch (to VDE 0113)<sup>1)</sup> or a disconnection facility (to VDE 0100)<sup>2)</sup> must be provided for the CC, EU and load power supply.
- For DC 24 V load circuits you require a load power supply with guaranteed electrical isolation. Non-regulated load power supplies must be provided with a capacitor (dimensioning: 250  $\mu$ F per 1 A load current). This means you must connect a capacitor in parallel to the output terminals.
- Electrical isolation by means of a transformer (to VDE 0113)<sup>1)</sup> Section 6.1.1 and VDE 0100<sup>2)</sup>) is recommended for load circuits for supplying external control devices with electromagnetic operating coils (e.g. more than 5).
- The circuits for the sensors and actuators can be used in groups.
- To protect against parasitic voltages, the subracks must be connected together with a large-area contact and low impedance.

1) VDE 0113 is equivalent to EN 60 204, IEC 204-1

2) VDE 0100 is equivalent to CENELEC HD 384.4.31 (IEC 364-4-41).



### 4.3.1 Power Supply for CCs, EUs and Process Peripherals from Grounded Battery or Grounded Power Supply Units

The ground of the internal supply voltages is connected to the subrack housing. Grounded operation provides the best noise immunity.

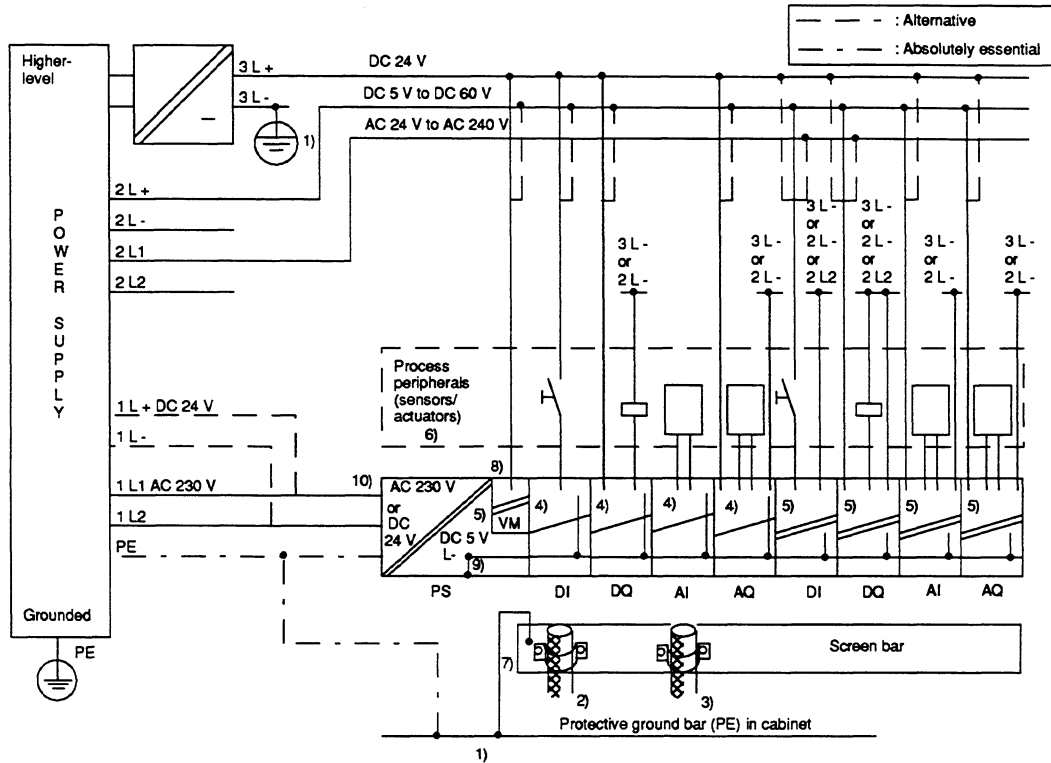


Fig. 6 Possible connections for sensors/actuators of the process peripherals to grounded power supply units

- 1) Housing potential (cabinet potential) = protective ground conductor.
  - 2) Use cable screen, if available, for digital modules. Provide screening with longer cables; connect at one end to cabinet inlet or connect at both ends.
  - 3) Connect cable screen at one end to cabinet inlet with analog modules or also at both ends; lead on up to module.
  - 4) Non-floating module.
  - 5) Floating module.
  - 6) Protective ground conductor required to housings of sensors and actuators.
  - 7) Connection cable with as large a cross-section as possible (black) > 16 mm<sup>2</sup>; if the screen is used as the protective ground conductor (green/yellow), connect at both ends.
  - 8) Only with S5-135U/155U series: monitoring of load voltage L+ (24 V DC).
  - 9) Non-removable connection between the internal ground of the supply voltages and the housing.
- Particularly important:**
- 10) **Electrical isolation is not available with the power supply unit 24 V/10 A (order no. 6ES5 955-3NA12); operation is only possible without problems on grounded power supply unit.**

### 4.3.2 Power Supply for CCs, EUs and Process Peripherals from Centrally Grounded Battery or Centrally Grounded Power Supply Units

If SIMATIC S5 programmable controllers are to be installed where a central grounding is available, then proceed as shown in Fig. 7. This is, however, not as immune to noise as the grounded system in Fig. 6.

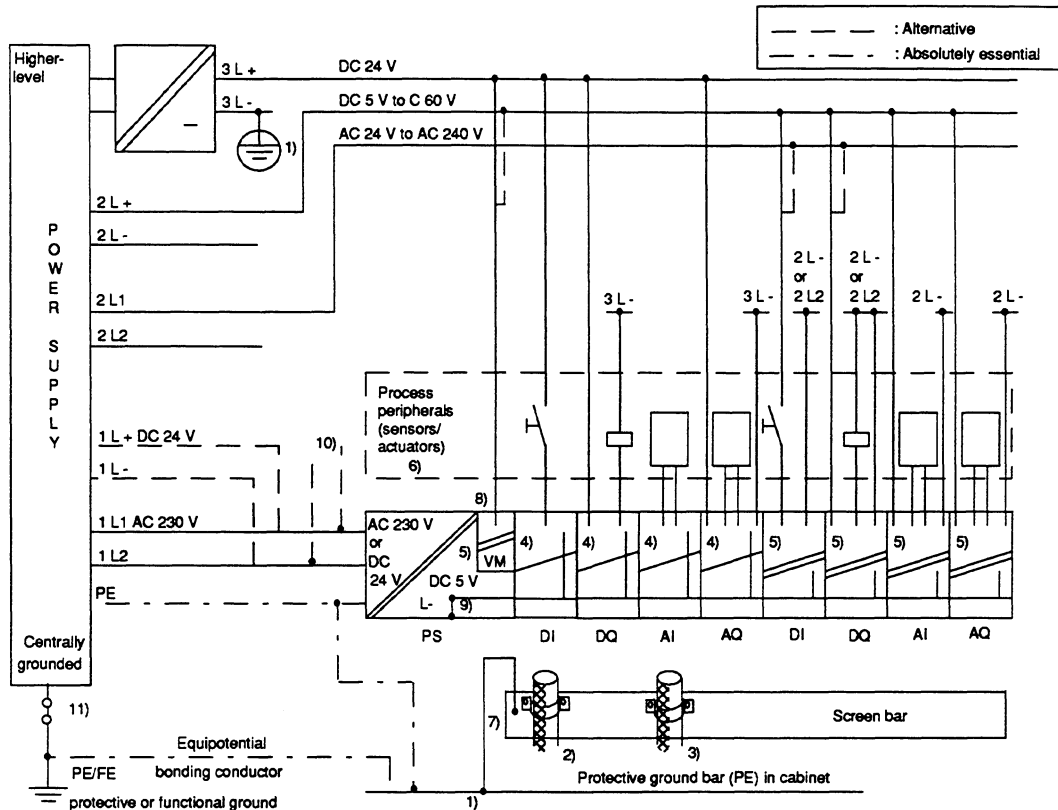


Fig. 7 Possible connections for sensors/actuators of the process peripherals to centrally grounded power supply units

- 1) Housing potential (cabinet potential) = protective ground conductor.
- 2) Use cable screen, if available, for digital modules. Provide screening with longer cables; connect at one end to cabinet inlet or connect at both ends.
- 3) Connect cable screen at one end to cabinet inlet with analog modules or also at both ends; lead on up to module.
- 4) Non-floating module.
- 5) Floating module.
- 6) Protective ground conductor required to housings of sensors and actuators; can be omitted for safely generated functional extra-low voltages.
- 7) Connection cable with as large a cross-section as possible (black) > 16 mm<sup>2</sup>.
- 8) Only with S5-135U/155U series: monitoring of load voltage L+ (24 V DC).
- 9) Non-removable connection between the internal ground of the supply voltages and the housing.

**Particularly important:**

- 10) **Electrical isolation is not available with the power supply unit 24 V/10 A (order no. 6ES5 955-3NA12); operation on a centrally grounded power supply unit is therefore not directly possible. Voltage supply required via 3L+/-.**
- 11) **Removable connection for test purposes.**

### 4.3.3 Power Supply for CCs, EUs and Process Peripherals from Non-Grounded Battery or Non-Grounded Power Supply Units

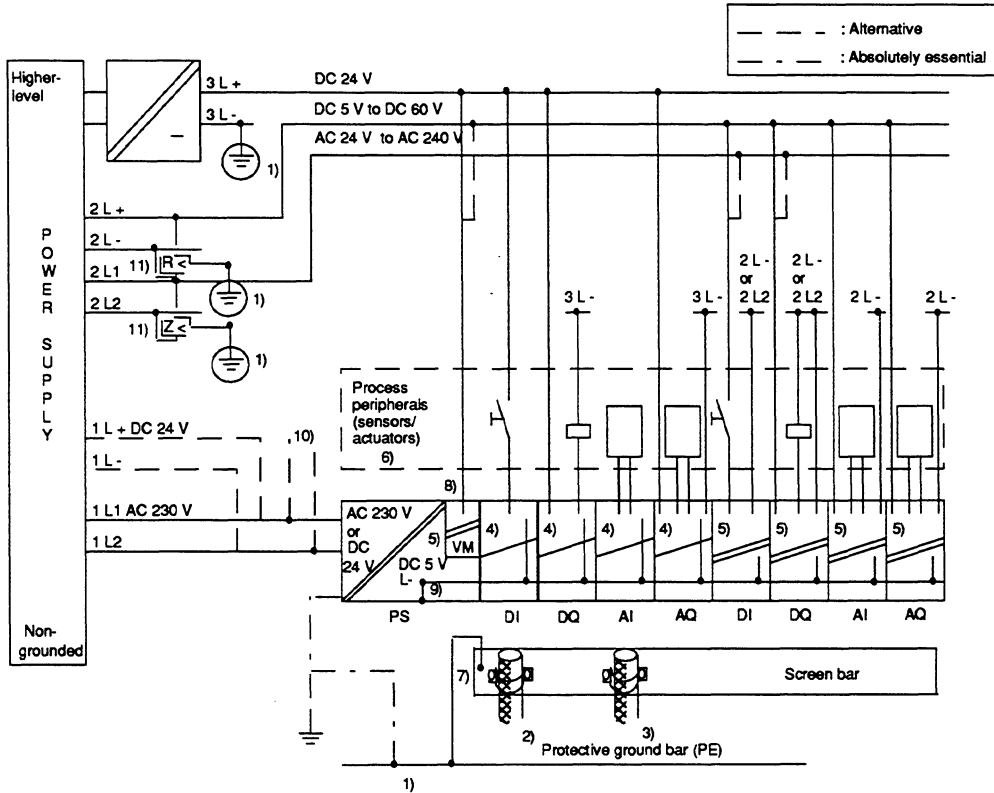


Fig. 8 Possible connections for sensors/actuators of the process peripherals to non-grounded power supply units

- 1) Housing potential (cabinet potential) = protective ground conductor.
- 2) Use cable screen, if available, for digital modules. Provide screening with longer cables; connect at one end to cabinet inlet or connect at both ends.
- 3) Connect cable screen at one end to cabinet inlet with analog modules or also at both ends; lead on up to module.
- 4) Non-floating module.
- 5) Floating module.
- 6) Protective ground conductor required to housings of sensors and actuators; can be omitted for safely generated functional extra-low voltages.
- 7) Connection cable with as large a cross-section as possible (black); if the screen is used as the protective ground conductor (green/yellow), connect at both ends.
- 8) Only with S5-135U/155U series: monitoring of load voltage L+ (24 V DC).
- 9) Non-removable connection between the internal ground of the supply voltages and the housing.

**Particularly important:**

- 10) **Electrical isolation is not available with the power supply unit 24 V/10 A (order no. 6ES5 955-3NA12); operation on a centrally grounded power supply unit is therefore not directly possible. Voltage supply required via 3L+/-.**
- 11) **Insulation monitoring equipment is required if dangerous conditions could result through double faults and/or with voltages > 42 V AC or 120 V DC. Only one insulation monitor is required per supply unit (to VDE 0113 Section 6.2.2).**

## 4.4 Load Power Supply from Two Power Supply Units

The design of the load power supply using two power supply units enables you to specifically disconnect parts of the process peripherals. The inputs and outputs of different modules can be assigned as a group to one power supply unit.

The supply to inputs and outputs of different modules from two power supply units is indicated below using two examples.

### 4.4.1 Non-floating Modules

In the case of non-floating input/output modules it must be ensured that the negative poles (L-) of the power supply units are connected to the reference potential (SIMATIC device/cabinet housing). This is necessary since the inputs are referred to ground.

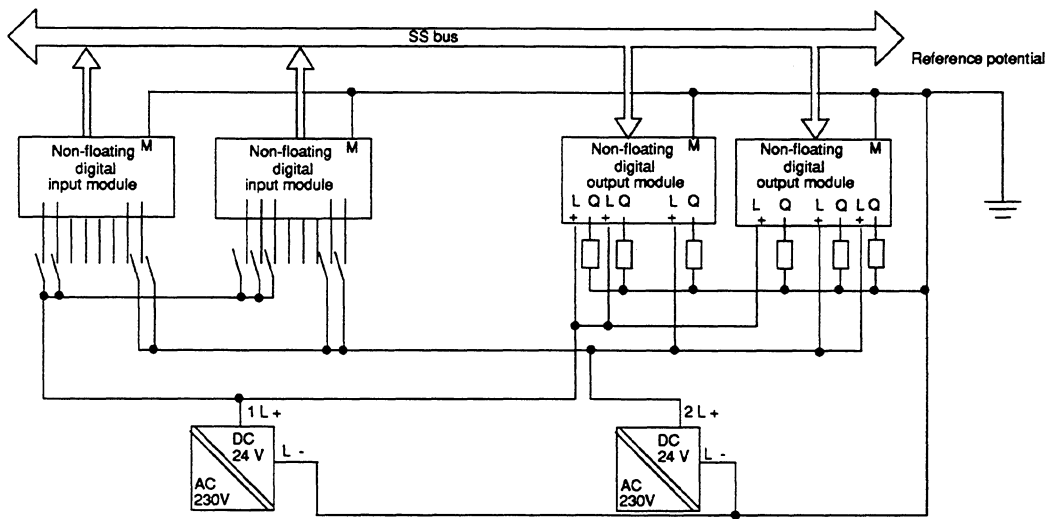


Fig. 9 Example of supply of non-floating peripheral modules from two power supply units

### 4.4.2 Floating Modules

In the case of floating modules, the inputs or outputs can be supplied from two power supply units by dividing into isolated groups.

Note that electrical isolation between the groups is lost as a result of the connection of inputs or outputs of two floating groups to one power supply unit.

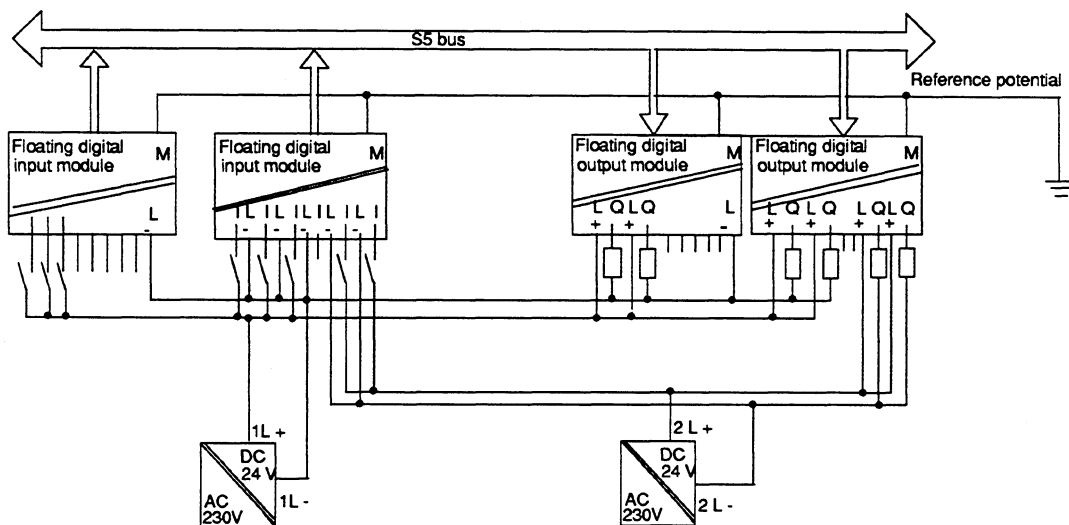


Fig. 10 Supply of floating peripheral modules from two power supply units

When using CCs and EUs with a mains connection, we recommend a Siemens power supply unit from the 6EV13.. series with electrical isolation as the power supply for the process peripherals (load voltage).

## 5 Wiring Layout

You can achieve a high noise immunity for your programmable controller by using a correct wiring layout. The measures required are described in the following sections.

### 5.1 Wiring Layout Inside a Cabinet

To ensure a correct layout of wiring inside cabinets, the wiring must be divided into the following groups:

Group A:   screened data lines (for PG, OP, SINEC L1, CP 525 etc.)  
              screened analog lines  
              screened signal lines for DC and AC voltages  $\leq 400$  V  
              non-screened lines for DC and AC voltages  $\leq 60$  V

Group B:   non-screened lines for DC and AC voltages  $> 60$  V and  $\leq 400$  V

Group C:   non-screened lines for DC and AC voltages  $> 400$  V and  $\leq 1$  kV

Route all wiring groups separately in the cabinet. Separately means that the wiring is routed in

- separate cable ducts
- separate wiring bundles with approx. 10 cm clearance.

When laying screened lines (e.g. analog lines) make a large-area contact of the screen to a cable clamping rail at the inlet to the cabinet and connect the screen further to the final point without an interruption (see Section 6.4).

## 5.2 Wiring Layout Outside Cabinets

- Route the cables outside cabinets and within buildings on metal cable trays. Make a conductive connection between the ends of two adjacent cable trays and connect these to ground at distances of 20 to 30 m.

The following may be routed on the same cable trays (cable routes, gutters, channels):

- cables from group A and
- cables from group B with approx. 10 cm clearance.

Route cables in group C on separate cable trays (cable routes, conduit).

- Always screen analog lines.
- Non-screened cables (e.g. signal lines, power supply lines) must be routed with as large a clearance from sources of interference (contactor, transformer, motor, electric welding unit) as possible.
- Signal lines and associated equipotential bonding lines should be routed with the smallest possible distance from one another and on the shortest path.
- Lines between the programmable controller and sensors/load should be installed whenever possible without breaks. If a break in the line is unavoidable, screen the terminal block e.g. with a metal box making large-area contact to a screen bar.
- Route associated single lines (e.g. forward and return lines, power supply cables) as close as possible to one another. If possible these lines should be twisted.

**Note:**



Signal lines and power cables up to 1 kV must be routed separately but can be routed in parallel. A minimum clearance of 10 cm must be observed. The clearance should be increased proportionally with higher voltages, and the safety regulations must be observed (e.g. IEC 664/664A).

### 5.3 Wiring Layout Outside Buildings

- If you route cables outside buildings, a double-screened cable must always be used for analog and data signal transmissions.  
The following must be observed when routing double-screened cables:
  - connect the outer screen to ground at both ends
  - only connect the inner screen at one end to the receiver side.
- Ensure that the equipotential bonding is sufficient. Connect an equipotential bonding conductor if necessary.
- The lightning protection and grounding regulations must be observed.

### 5.4 Equipotential Bonding

Different potentials can occur between different parts of your plant (e.g. different power supplies). These differences can be reduced by laying equipotential bonding lines to ensure the correct functioning of electronic components.

Keep the following points in mind when laying an equipotential bonding line:

- The effectiveness of equipotential bonding is directly related to the impedance of the line (less impedance - greater effectiveness). This means that the connection required for equipotential bonding must have not only a low ohmic resistance but also as small an inductance as possible (achieved by keeping line lengths short).
- If screened signal lines with the screens grounded at both ends are required between parts of the plant, the impedance of the additional equipotential bonding line must not exceed a maximum of 10% of the screen impedance.
- The cross-sectional area of the equipotential bonding line must be selected for the max. equalizing currents.
- The equipotential bonding line must be laid so that loops (e.g. between equipotential bonding line and signal lines) cover as small an area as possible.
- The equipotential bonding line must make large-area contact with ground or chassis (see Section 6.4)



## **6 Cabinet Wiring and Design with Respect to EMC**

EMC: electromagnetic compatibility (EMC) is understood to be the ability of an electric device to function without faults in a defined electromagnetic environment without influencing other devices in the environment.

Measures to guarantee EMC must already be made when designing and wiring the individual components in cabinets. The interfering environment must not be ignored if fault-free functioning of the programmable controller and wiring is to be obtained.

The measures required to guarantee EMC, as well as an example of a cabinet design as concerns EMC, are described in the following sections. The check list at the end of this section serves as an aid for checking the EMC-compatible design of your cabinet.

The following section as well as Section 5.1, Wiring Layout Inside a Cabinet, must be observed when designing your cabinet to guarantee EMC. These sections handle the subjects:

- Grounding of all inactive metal components
- Wiring layout in the cabinet
- Shielding of devices and cables
- Use of special interference-suppression measures.

## 6.1 Grounding of Inactive Metal Components

An important factor which contributes towards interference-free operation is consistent grounding. Grounding is understood to be the electrical connection of all inactive metal components (VDE 0160). Large-area grounding must always be used.

Large-area grounding means:

- Ground all conducting parts.  
These include subracks, cabinet members, cabinet panels, cabinet doors, screen bars, filter housings.

Measures to be observed when grounding:

- Make all ground connections with a low impedance.
- Connect all metal parts with a large-area contact.
- Use ground straps for the connection. Metallic wire mesh made of tin-plated copper strands is suitable as the ground strap. It should be kept as short as possible. The surface area of the ground straps is decisive, and not the cross-section, because of the high-frequency noise pulses discharged.
- Make the screw connections using NOMEL contact washers<sup>1)</sup>.

### NOMEL contact washers

Assemble contact washers such that the teeth bite into the part to be screwed and thus generate a metallic contact.

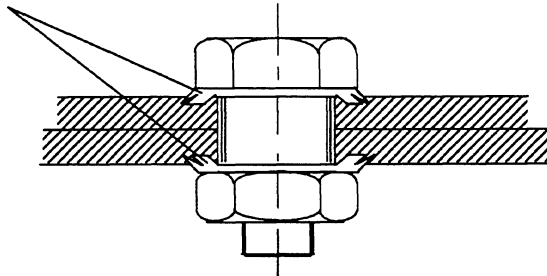


Fig. 11 NOMEL contact washers

<sup>1)</sup> Contact washer Siemens standard 70093 available from  
- Siemens ANL A443 Werkzeug 8520 Erlangen  
- Teckentrup GmbH und Co. KG, Postfach 120, D-5974 Herscheid 2,  
- NOMEL S.A. Tour Franklin, Cedex 11, F-92081 Paris.  
- or from your local Siemens representative

## 6.2 Shielding of Devices and Cables

Shielding is a way of attenuating (dampening) magnetic, electric or electromagnetic interferences. Shielding can be divided into:

### Device shielding

Cabinets and housings must be incorporated into the measures for shielding the programmable controllers. The following must be observed:

- Cabinet enclosures such as side panels, rear walls, roof and floor panels must be connected sufficiently often with a low impedance in the case of an overlapping arrangement (connection interval e.g. 50 mm).
- Doors must additionally be connected to the cabinet ground. Use at least 2 ground straps.
- If sources of strong interference are present in the cabinet (transformers, cables to motors etc.), these must be isolated from sensitive electronics areas by metal partitions (steel, highly permeable material, e.g. mu-metal). The panels must be screwed several times to the cabinet ground with a low impedance.

The central grounding point must be connected to the protective ground conductor (grounding bar) with a low impedance and a Cu conductor  $\geq 16 \text{ mm}^2$  as short as possible.

### Cable screening

Screened cables must be connected at both ends to the grounding bar with a large-area contact and if possible directly at the cabinet inlet. Good attenuation of all conducted frequencies can only be achieved by connecting at both ends.

The following must be observed when handling the screen:

- Use metal cable clamps to secure the braided screens with a large-area contact.
- Avoid the use of cables with foil screens since the foil can be easily damaged by tension or pressure when fitting, thus leading to a poorer screening effect.

**Note:**



An equalizing current may flow via the screen connected at both ends in the case of variations in the ground potential. Use an additional equipotential bonding conductor in this case (see Section 5.4 Equipotential Bonding).

In certain cases the screen can also be connected at only one end. Only the lower frequencies are then attenuated. Connection of the screen at one end may be more favorable if:

- An equipotential bonding conductor cannot be laid
- Analog signals (several mV or  $\mu\text{A}$ ) are transmitted.

Interferences on cable screens are discharged to ground via the grounding bar and the equipotential bonding conductor. A low-impedance path to ground for the interfering currents must be provided so that these discharged currents do not produce a source of interference themselves:

- Tightly connect the screws of cable plugs, modules and equipotential bonding conductors.
- Protect the contact surfaces of equipotential bonding conductors and ground lines from corrosion.

## 6.3 Use of Special Noise Suppression Measures

### Connection of inductors

Provide suppression (e.g. using RC elements, varistors or free-wheeling diodes) for inductors installed in the same cabinet (e.g. contactor and relay coils) not activated by SIMATIC S5 modules.

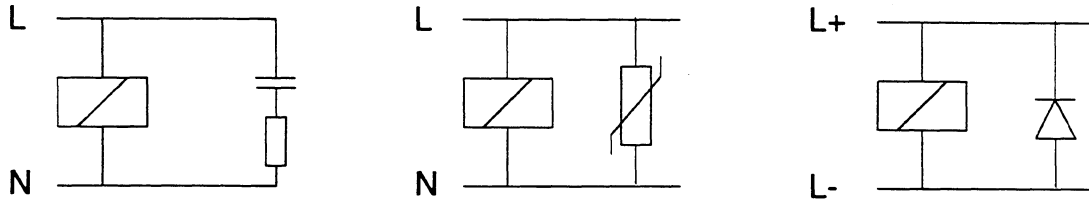



Fig. 12 Wiring inductors (example)

If further contacts are connected in series to SIMATIC outputs, the SIMATIC internal fusing is not effective. In such cases the inductor must be fused directly.

### Protection against electrostatic discharge

Use metal housings or cabinets that are closed in at all sides to protect devices and modules against electrostatic discharge. Connect these housings or cabinets to the grounding point where you set them up so as to form a good contact.

|   |   |
|---|---|
| <b>Caution:</b>   |   |
|  | If you must work on the system with the cabinet open, follow the guidelines to protect electrostatically sensitive devices and modules (ESD). |

The interference resistance is always reduced when the cabinet is open.

### Mains power connection for programmers

Provide a grounded socket in each cabinet to supply power for a programmer. The sockets should be connected to the distribution board to which the protective ground conductor of the cabinet is also connected.

### **Cabinet illumination**

Do not use fluorescent lamps for the cabinet illumination since these generate interferences. If you must use fluorescent lamps, take the precautions shown in Fig. 13 LINESTRA<sup>®</sup> lamps are more suitable.

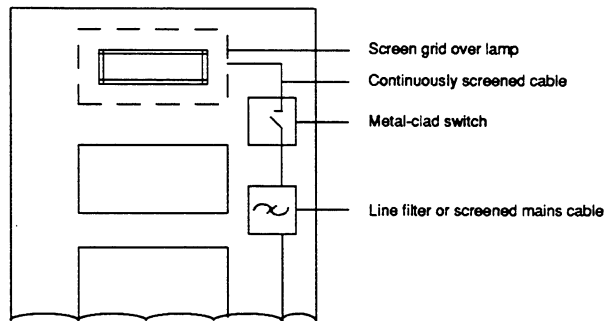


Fig. 13 Measures to suppress noise from fluorescent lamps in a cabinet

### 6.4 Example of an EMC-compatible Cabinet Design

The example of a cabinet design shown in Fig. 14 - taking into consideration EMC - shows the grounding of all inactive metal components and the connection of screened cables. This example only applies to grounded operation. Observe the points listed in Fig. 14 during installation.

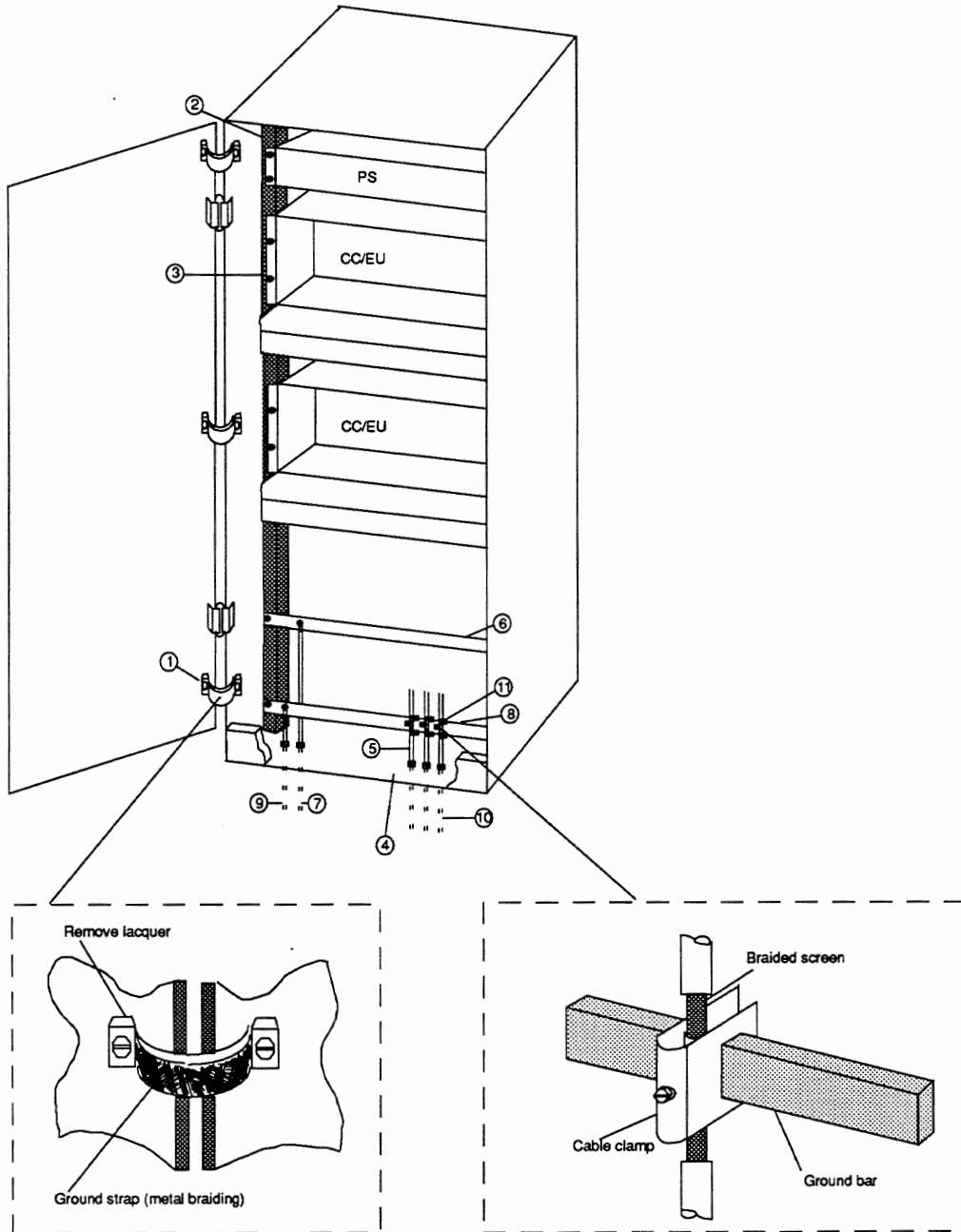


Fig. 14 Example of an EMC-compatible cabinet design

**Re 1. Ground straps**

All inactive metal components (e.g. cabinet doors and supporting panels) must be connected using ground straps if large-area metal-metal connections are not present. Metallic wire mesh made of tin-plated copper strands is suitable as the ground strap. It should be kept as short as possible, with a ratio between the length and width of less than 3 to 1.

**Re 2. Cabinet members**

The cabinet members must be connected to the cabinet housing with a large-area contact (metal-metal connection).

**Re 3. Mounting bracket**

A large-area metal-metal connection must be made between the cabinet member and mounting bracket.

**Re 4. Base panel**

A large-area metal-metal connection to the cabinet housing must be guaranteed.

**Re 5. Cable screwed glands**

Unused cable screwed glands must be closed using blanking plates in the case of closed cabinets with heat exchangers.

**Re 6. Equipotential bonding bar**

The bar must be connected to the cabinet members with a large-area contact (metal-metal connection).

**Re 7. Equipotential bonding conductor**

The conductors must be connected to the equipotential bonding bar.

**Re 8. Ground bar**

This serves as the central grounding point of the cabinet and must be connected to the cabinet members with a large-area contact (metal-metal connection). The ground bars must be connected to the external central grounding point to guarantee discharging of interfering and fault currents. It can additionally be used to connect screened cables.

**Re 9. Cable from central grounding point**

The cable must be connected to the grounding bar with a large-area contact.

**Re 10. Signal cables**

The screen of screened signal cables must be connected to the grounding bar with a large-area contact using cable clamps or to an additional screen bar connected with a large-area contact, and then routed further to the end point (e.g. I/O module) without interruption.

**Re 11. Cable clamp**

The cable clamp must enclose the braided screen over a large area.



## 6.5 Checklist for EMC-compatible Cabinet Design

| EMC measures  | Remarks |
|---|---------|
| <b>Connection of inactive components</b> (Section 6.1)  |         |
| Are all inactive metal components connected together with a large-area contact and low impedance, and grounded?   |         |
| Is there a sufficient connection to the central grounding point?  |         |
| Screw connections made using NOMEL contact washers?   |         |
| Particularly check the connections to: <ul style="list-style-type: none"> <li>• Subracks</li> <li>• Cabinet members</li> <li>• Cabinet bar</li> <li>• Filter housing</li> </ul> |         |
| <b>Equipotential bonding</b> (Section 5)  |         |
| With a spacially separated design, check the routing of the equipotential bonding conductor   |         |
| <b>Device screening</b> (Section 6.2)   |         |
| All cabinet components provided with contacts at sufficient intervals?  |         |
| Doors connected to the cabinet body using ground straps?  |         |
| Are only metallic device plugs used?  |         |
| <b>Cable screening</b> (Section 6.2)  |         |
| Are all analog cables screened?<br>Are screens connected at both ends?  |         |
| Are cable screens connected to ground bar or screen bar at cabinet inlet?   |         |
| Are cable screens connected via cable clamps with a large-area contact, completely enclosed and with a low impedance?   |         |
| <b>Inductors</b> (Section 6.3)  |         |
| Are isolating panels used in event of magnetic influences from inductors?   |         |
| Are all coils of contactors connected to RC elements?   |         |

Table 2 Checklist for EMC-compatible cabinet design

| EMC measures   | Remarks |
|--|---------|
| <b>Cable routing</b> (Section 5)   |         |
| Cabling divided into groups?   |         |
| Power supply cables (230 V) and signal cables routed in separate ducts or bundles? |         |
| Complete cabling introduced into cabinet at one position?                          |         |
| Signal cables routed close to grounded surface?                                    |         |
| Forward and return lines of individually routed conductors twisted if possible?    |         |

Table 2 Checklist for EMC-compatible cabinet design (continued)

## 7 Framework and Wall Mounting of SIMATIC S5 Controllers

If you operate your SIMATIC controllers in an environment which is free from interferences to the greatest possible extent, you can fit the central controllers and expansion units on a framework or directly on a wall.

The following must be observed:

- A reference surface made of sheet-steel should be provided to improve the deviation of interfering currents conducted via the inlet cables. This reference surface must be at least 480 mm x 250 mm large and connected to the central grounding point. If you use screening or cable clamping rails, space must be provided for these on the reference surface. In the case of framework mounting, the metal frame serves as the reference surface.
- Fit the screen bar or cable clamping bar to this reference surface or to the framework. Ensure that the connection between the rails and the reference surface or framework is made with a large-area contact and low impedance (metal-metal connection).
- Connect all inactive metal components together with a large-area contact and low impedance. Inactive metal components are: subrack, power supply, reference surface, screen bar, protective ground bar.
- Also observe the points for the wiring layout (see Section 5).

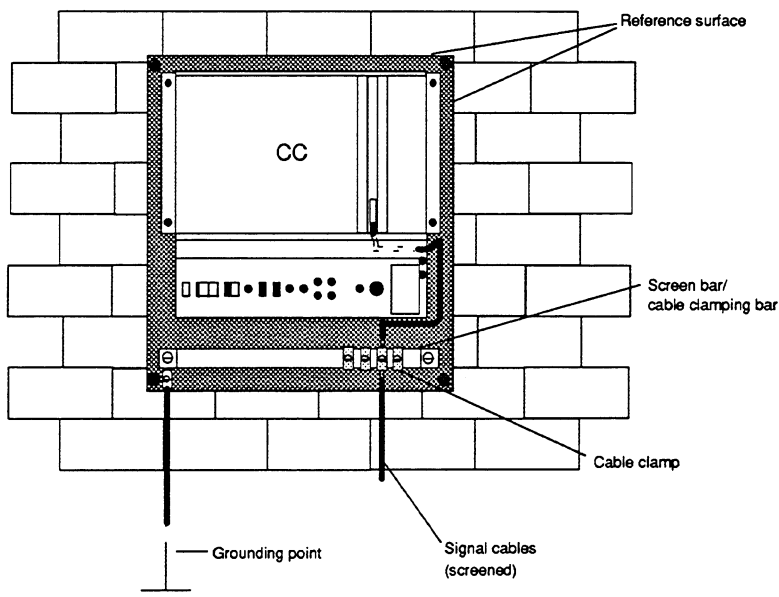


Fig. 15 Wall mounting



**Note:**

If you mount subracks from the S5-135U/155U series on a wall, provide a minimum clearance from the rear panel of  $\geq 30$  mm. This is necessary to guarantee heat exchange on the heat sink of the power supply.



**Note:**

When using radio telephones the field strength must not exceed 3 V/m at the programmable controller.

Owing to unknown values such as output power and frequency range, radio telephones should only be used when a certain safety clearance from PLCs not installed in cabinets is maintained.

## 8 Lightning Protection Measures

If cables and lines for SIMATIC S5 devices are laid outdoors, the lightning protection regulations must be adhered to.

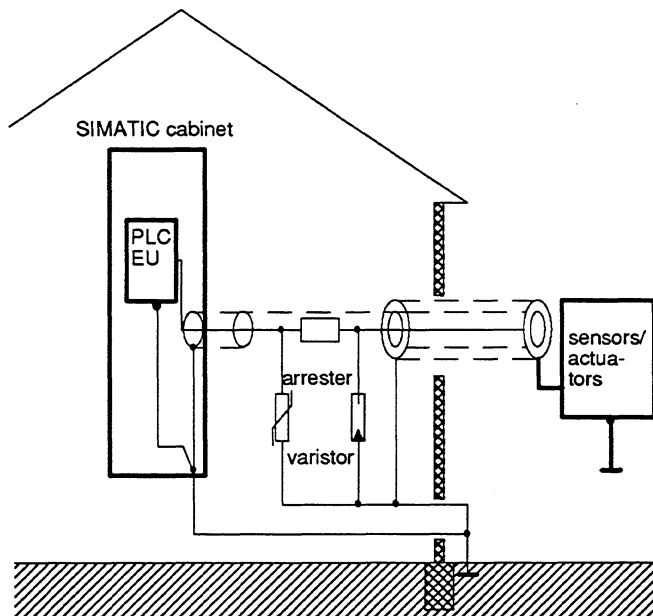


Fig. 16 Arrangement of lightning protection elements

Protect signal lines from overvoltages as follows:

- varistors
- or
- inert gas-filled lightning arresters

Install these protective elements

- as close as possible to the point of entry into the building
- before cables enter the cabinet



**Note:**

Lightning protection must be adapted to each situation individually. Please contact your local Siemens representative if you require advice.

## 9 Safety Measures

When configuring systems that have programmable controllers - as is the case with contactor equipment - follow the relevant regulations: CENELEC HD 384.4.41 (IEC 364-4-41) "Electrical installations of buildings" and also EN 60204 (European standard, corresponds to IEC 204-1), "Electrical equipment of industrial machines" (VDE 0113).

Pay special attention to the following points:

- Prevent conditions that could endanger or injure people or which could damage machines and material.
- When power is restored after a power failure or after EMERGENCY STOP units are released, machines must not be able to restart automatically.
- When a programmable controller malfunctions, commands from EMERGENCY STOP units and from safety limit switches must remain effective under all conditions. These safety measures must have a direct effect on the actuators in the power circuit independent of the programmable controller.
- When EMERGENCY STOP units are activated, safety must be guaranteed for people and systems as follows:

**Note:**

As operator of the system, you are responsible for the measures described in the section "Safety Measures" to avoid dangerous situations.

### 9.1 Protection against Indirect Contact

Parts which can be touched must not carry dangerous currents even in the event of a fault. They must be incorporated into the protective measures against dangerous currents.

This requirement is satisfied if all metal parts which can be touched and which could be dangerous in the event of a fault are safely connected electrically to the protective ground conductor (PE). The max. permissible resistance between the protective ground conductor and the part to be protected is 0.5  $\Omega$ .

# SIEMENS

## SIMATIC S5

155U Central Controller  
for the S5-155U and S5-155H

---

Instructions

C79000-B8576-C380-05

---

## **Preface**

This manual contains the hardware description and installation and maintenance procedures for the central controller 155U (6ES5 155-3UA11 and 6ES5 155-3UA21). This manual also provides everything you need to know about operation, maintenance and technical specifications.

This manual is intended for engineers, programmers, and maintenance personnel.

If you have any questions about the S5-155U central controller not answered in this manual, please contact your local Siemens representative.





## How to Use This Book

This section discusses information that may be helpful as you use this book.

The main information you will find in:

- Section 1.2.1, Possible Configurations
- Chapter 2, Installation / Dimensions of the S5-155U Central Controller
- Section 4.5.3, Checklist for Starting Up

The individual chapters offer you the following:

- **Chapter 1: Technical Description**  
This chapter discusses the application of the S5-155U programmable controller and describes its central controller. It includes details on possible configurations and possible links with expansion units.
- **Chapter 2: Installation of the S5-155U Central Controller**  
This chapter describes the installation procedure for the central controller, including its integrated power supply and 15-V supplementary module.
- **Chapter 3: Wiring Connections on the Power Supply Unit of the S5-155U Central Controller**  
This chapter describes all connections and explains how to set the installed fan and battery monitoring. It also includes recommendations on wiring for fan and temperature monitoring on the programmable controller.
- **Chapter 4: Operation of the S5-155U Central Controller**  
This chapter discusses the commissioning and the operation requirements. It explains the LEDs and operating elements on the power supply unit, jumper locations, and the functions of the alarm relays in the power supply unit.
- **Chapter 5: Maintenance of the S5-155U Central Controller**  
This chapter explains how to change the modules and the back-up battery. It also provides information on the connector pin assignments of the bus PCB including the interrupt signals.
- **Chapter 6: Technical Data of the S5-155U Central Controller**  
This chapter lists the technical data for the central controller, including its integrated power supply unit and 15-V supplementary module. It informs you about device safety, climatic and mechanical ambient conditions and interference immunity.
- **Index**  
The index contains an alphabetical list of key words and subjects covered in this book and their corresponding page numbers.
- **Remarks Form**  
The remarks form is provided for your comments and recommendations.

## Training

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

## Reference Material

The following books that support the S5-155U system are recommended:

- *Catalog ST 54.1: S5-135U, S5-155U and S5-155H Programmable Controllers (Order No. E86010-K4654-A111-A6-7600)\**

Programmer manuals:

- *S5-135U (CPU 928B) (Order No. 6ES5 998-2UL22)\**
- *S5-135U (CPU 928) (Order No. 6ES5 998-1UL23)\**
- *S5-135U (S and R Processor) (Order No. 6ES5 998-0UL22)\**
- *U Periphery (Order No. 6ES5 998-0PC22)\**
- *PG 685 Programmer (Order No. 6ES5 885-0SC21)\**
- *PG 710 Programmer (Order No. 6ES5 814-0MC21)\**
- *PG 730 Programmer (Order No. 6ES5 834 0FC21)\**
- *PG 750 Programmer (Order No. 6ES5 886-0FC21 for Processor 386)\* (Order No. 6ES5 886-0FC22 for Processor 486)\**
- *PG 770 Programmer (Order No. 6ES5 887-0FC21)\**
- *STEP 5 Programming Package for Personal Computers (Order No. 6ES5 896-0SC21)\**
- *You will find an introduction to programming with STEP 5, as well as an explanation of how to work with the S5-155U programmable controller and its I/O modules in the following book:*

*Automating with the SIMATIC S5-155U  
by Hans Berger  
Siemens AG, ISBN 3-8009-1562-6*

---

\* Order this book from your local Siemens representative.

## Contents

|          |  |              |
|----------|--|--------------|
|          | <b>Preface .....</b>   | <b>0 - 1</b> |
|          | <b>How to Use This Book.....</b>   | <b>0 - 3</b> |
| <b>1</b> | <b>Technical Description of the S5-155U Central Controller .....</b>                       | <b>1 - 1</b> |
| 1.1      | Application.....   | 1 - 1        |
| 1.2      | Design .....   | 1 - 3        |
| 1.2.1    | Possible Configurations of the S5-155U Central Controller .....                            | 1 - 5        |
| 1.2.2    | Possible Configurations of the S5-155H Programmable Controller .....                       | 1 - 6        |
| 1.2.3    | Addressing the I/O Modules.....  | 1 - 7        |
| 1.3      | Device Configuration of the S5-155U Programmable Controller .....                          | 1 - 8        |
| 1.4      | Interfacing Between Central Controllers and Expansion Units .....                          | 1 - 8        |
| 1.4.1    | Central Interfacing.....   | 1 - 9        |
| 1.4.2    | Distributed Interfacing .....  | 1 - 10       |
| 1.4.3    | Examples of Interfacing Possibilities.....   | 1 - 12       |
| 1.5      | Mode of Operation .....  | 1 - 13       |
| 1.5.1    | Single/Multiprocessor Operation .....  | 1 - 13       |
| 1.5.2    | Unit Interfacing .....   | 1 - 14       |
| <b>2</b> | <b>Installation of the S5-155U Central Controller.....</b>                                 | <b>2 - 1</b> |
| 2.1      | Installing the Central Controller.....   | 2 - 1        |
| 2.2      | Installing the Power Supply Unit .....   | 2 - 3        |
| 2.2.1    | Installing the 15-V Supplementary Module .....   | 2 - 3        |
| 2.3      | Installation of the Modules .....  | 2 - 4        |
| 2.3.1    | Connections to the CPUs, CPs and Interface Modules .....                                   | 2 - 4        |
| 2.3.2    | Connections to the I/O Modules.....  | 2 - 4        |
| <b>3</b> | <b>Wiring Connections on the Power Supply Unit of the S5-155U Central Controller .....</b> | <b>3 - 1</b> |
| 3.1      | Connections of the Power Supply Units.....   | 3 - 1        |
| 3.2      | Recommended Wiring for Fans and Temperature Monitoring .....                               | 3 - 2        |
| 3.2.1    | Setting the Fan Monitoring .....   | 3 - 3        |
| 3.2.2    | Setting the Back-Up Battery Monitoring .....   | 3 - 4        |

|          |  |              |
|----------|--|--------------|
| <b>4</b> | <b>Operation of the S5-155U Central Controller.....</b>  | <b>4 - 1</b> |
| 4.1      | General Notes on the Power Supply Unit .....   | 4 - 1        |
| 4.2      | Operating and Display Elements of the Power Supply Unit .....  | 4 - 3        |
| 4.3      | Functions and Locations of Jumpers on the Power Supply Unit<br>6ES5 955-3xyy .....                         | 4 - 4        |
| 4.4      | Power Supply Behaviour in the Event of Faults .....  | 4 - 6        |
| 4.5      | Start-Up and Functional Test<br>(only for S5-155U Programmable Controller).....                            | 4 - 7        |
| 4.5.1    | Start-Up with Single Processor Operation .....   | 4 - 8        |
| 4.5.2    | Start-Up with Multiprocessor Operation<br>(only for S5-155U Programmable Controller).....                  | 4 - 9        |
| 4.5.3    | Checklist for Starting Up .....  | 4 - 10       |
| <b>5</b> | <b>Maintenance of the S5-155U Central Controller/Pin Assignment.....</b>                                   | <b>5 - 1</b> |
| 5.1      | Removing and Inserting S5 Modules .....  | 5 - 1        |
| 5.2      | Removing and Inserting Power Supply Units.....   | 5 - 2        |
| 5.3      | Replacing the Back-Up Battery and the Fans .....   | 5 - 3        |
| 5.4      | Pin Assignments of the Power Supply Unit.....  | 5 - 5        |
| 5.5      | Pin Assignment of the Bus PCB .....  | 5 - 6        |
| 5.6      | Pin Designation of the Interrupt Signals on the Bus PCB<br>(only for S5-155U Programmable Controller)..... | 5 - 11       |
| <b>6</b> | <b>Technical Data of the S5-155U Central Controller .....</b>  | <b>6 - 1</b> |
| 6.1      | Power Supply Unit 6ES5 955-3LF12 .....   | 6 - 3        |
| 6.2      | Power Supply Unit 6ES5 955-3NF11 .....   | 6 - 5        |
|          | <b>Index.....</b>  | <b>I - 1</b> |

**Figures**

|    |  |      |
|----|--|------|
| 1  | S5-155U central controller .....   | 1-3  |
| 2  | Device configuration of the S5-155U programmable controller .....  | 1-8  |
| 3  | Example of central design with S5-155U central controllers<br>and EG-183U / S5-184U expansion units .....  | 1-9  |
| 4  | Example of a distributed configuration up to max. 600 m with<br>S5-155U central controller and EG-183U/184U/187U expansion units,<br>ER 701-1/701-3 subracks ..... | 1-10 |
| 5  | Device dimensions of the S5-155U central controller .....  | 2-1  |
| 6  | Different ways of fixing mounting brackets .....   | 2-2  |
| 7  | Mounting location of the 15-V supplementary module .....   | 2-3  |
| 8  | Connections of the power supply unit .....   | 3-1  |
| 9  | Recommended connections for fan/temperature monitoring<br>when using S5-155U CC and EG-183U EUs.....   | 3-2  |
| 10 | Operating and display elements of the power supply units.....  | 4-3  |
| 11 | Power supply unit 6ES5 955-3LF12 .....   | 4-5  |
| 12 | Starting up.....   | 4-7  |
| 13 | Battery compartment.....   | 5-3  |
| 14 | Plug X1, view from rear of device .....  | 5-5  |
| 15 | Plug X2, view from rear of device .....  | 5-5  |

**Tables**

|    |  |             |
|----|--|-------------|
| 1  | Assignment of power supply unit and S5-155U .....                | 1-1         |
| 2  | Possible configurations for the S5-155U central controller ..... | 1-5         |
| 3  | Possible configurations for the S5-155H central controller ..... | 1-6         |
| 4  | Examples of interfacing possibilities .....                      | 1-12        |
| 5  | Standard slots in ES 902 = 15.24 .....                           | 2-4         |
| 6  | 6ES5 955-3LF12 power supply unit jumper assignment .....         | 4-4         |
| 7  | Fault message and reaction of the power supply unit.....         | 4-6         |
| 8  | Pin assignment of the bus PCB .....                              | 5-6 to 5-10 |
| 9  | Pin designation of the interrupt signals .....                   | 5-11        |
| 10 | Technical data.....  | 6-1 to 6-6  |

**NOTE**

These instructions do not cover all details or variations in equipment or provide for every circumstance that can arise with installation, operation, or maintenance. If you want further information or if particular problems arise that are not covered sufficiently for your purposes, contact your local Siemens sales office.

The contents of this manual shall not become part of or modify any prior or existing agreement, commitment, or relationship. The sales contract contains the entire obligation of Siemens. The warranty contained in the contract between the parties is the sole warranty of Siemens. Any statements contained herein do not create new warranties or modify the existing warranty.

This description applies for the S5-155U programmable controller with the following power supply units:

| Order number of the central controller with power supply unit <sup>1)</sup> | Order number of integrated power supply unit | Technical data   |
|---|--|--|
| 6ES5 155-3UA11  | 6ES5 955-3LF12                               | 230 V/5 V/40 A<br>IN: 230 V/120 V~<br>OUT: 5 V-/40 A<br>24 V-/2,8 A<br>15 V-/... <sup>1)</sup> |
| 6ES5 155-3UA21  | 6ES5 955-3NF11                               | 24 V/5 V/40 A<br>IN: 24 V-<br>OUT: 5 V-/40 A<br>24 V-/2,8 A<br>15 V-/... <sup>1)</sup>         |

IN = INPUT (primary)

OUT = OUTPUT (secondary)

1) A 15-V module can be inserted into all power supply units. This is necessary if you use a CP 535 or CP 143 communications processor. The total current of the 24 V DC and 14 V DC supplies must not exceed the maximum current of 0.8 A or 2.8 A.

Table 1 Assignment of power supply unit and S5-155U

## 1 Technical Description of the S5-155U Central Controller

### 1.1 Application

The SIMATIC S5-155U programmable controller is a versatile multiprocessor unit for automation tasks in the top performance range.

The standardized instrument technology, the modular design of the units and the expansion facilities mean that the S5-155U programmable controller can be easily adapted to the respective automation tasks. It can be configured according to your requirements. The system provides you with various expansion facilities (e.g. S5-185U expansion unit), communications facilities (e.g. SINEC H1) and a range of operation, monitoring and programming devices of varying performance.

With the S5-155U programmable controller you can solve the following automation tasks simply and economically:

- Open-loop control
- Closed-loop control and computing
- Communication
- Operation and monitoring.

The controller is thus suitable for:

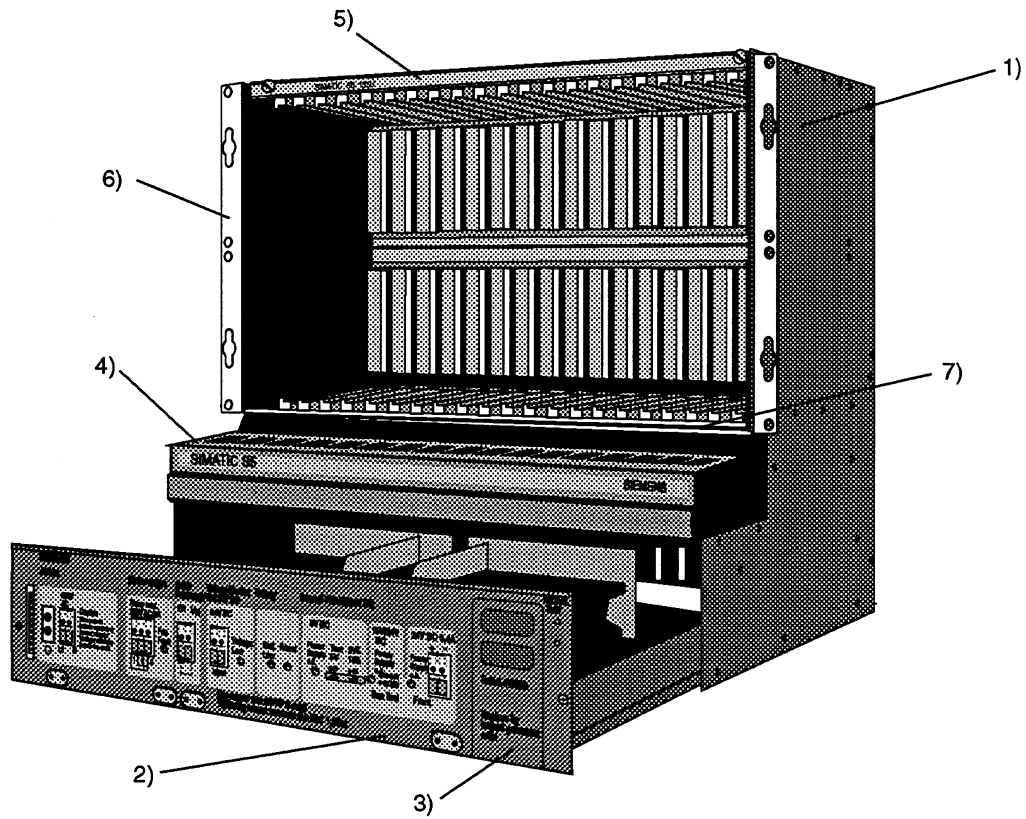
- Machine controls
- Process automation and
- Process monitoring.

The programming language is STEP 5 with the following methods of representation:

- Control system flow chart - CSF
- Ladder diagram - LAD
- Statement list - STL
- Higher-level sequence diagram - GRAPH 5
- Additionally the CPU 946/947 can be programmed in the programming language C.



## 1.2 Design



- 1 Housing with 21 module slots
- 2 Power supply unit with fans
- 3 Plug-in for back-up battery
- 4 Cable duct
- 5 Locking rail
- 6 Mounting bracket
- 7 Rail for individual locking

Fig. 1 S5-155U central controller

## **Housing**

The housing consists of screwed sheet-steel sections with ventilation openings at the top and bottom, as well as aluminium parts. The sheet-steel sections are chromium-plated, the aluminium locking rails are tin-plated. The housing contains the bus PCB which serves to connect the modules electrically. All slots have guide rails to ensure correct connection of the modules. At the top of the housing there is a locking bar to lock all modules at once. Modules with individual locking mechanisms can be secured using the bottom rail. A cable duct for incoming and outgoing signal cables is located at the front of the housing.

## **Power supply unit**

The power supply unit with its fans is accommodated in a tier at the bottom in the housing. The input voltage is either 24 V DC or 230/120 V AC depending on the type of power pack used. An internal selector is present for adaptation with 230/120 V AC.

The central controller is available with two types of power supply

- primary 120/230 V AC  
secondary +5 V/40 A, +24 V/2.8 A
- primary 24 V DC  
secondary +5 V/40 A, +24 V/2.8 A

A 15-V submodule must be installed in the power supply unit when using the SINEC H1 modules CP 535 or CP 143.

The CPU 946/947 consists of several components:

- CPU 946 double-width
- CPU 947 single-width
- either one or two 355 memory modules with RAM or EPROM memory submodules

### 1.2.1 Possible Configurations of the S5-155U Central Controller


The following table shows the possible configurations for the 155U CC.

| Module   | Slots | 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 | 67 | 75 | 83 | 91 | 99 | 107 | 115 | 123 | 131 | 139 | 147 | 155 | 163 |    |    |
|--|-------|---|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|----|----|
| COOR C   |       | ■ |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 6ESS 923-3UC..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| CPU 946 1)   |       |   | ■  |    |    |    |    | ■  |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 6ESS 946-3UA..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| CPU 947  |       |   |    |    | ■  |    |    |    |    | ■  |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 6ESS 947-3UA..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 355 memory module                                    |       |   |    |    |    | ■  |    |    |    |    | ■  |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 6ESS 355-3UA..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| CPU 920 (M-processor)                                |       |   | ■  |    |    |    |    | ■  |    |    |    |    |    | ■  | ■   |     |     |     |     |     |     |     |    |    |
| 6ESS 920-3UA..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| CPU 922 (R-processor)                                |       |   |    |    |    |    |    | ■  |    |    |    |    |    | ■  | ■   |     |     |     |     |     |     |     |    |    |
| 6ESS 922-3UA..                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| CPU 928 ,928B 1)                                     |       |   | ■  | ■  |    |    |    | ■  | ■  |    |    |    |    | ■  | ■   |     |     |     |     |     |     |     |    |    |
| 6ESS 928-3UA../-3UB11                                |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 300-3 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     | ■   | ■   | ■   | ■  |    |
| 6ESS 300-3...  |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 301-3 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     | ■   | ■   | ■   | ■  |    |
| 6ESS 301-3....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 304-3 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     | ■   | ■   | ■  |    |
| 6ESS 304-3....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 300-5 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     | ■   | ■  |    |
| 6ESS 300-5....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 301-5 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     | ■   | ■  |    |
| 6ESS 301-5....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 302-3 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     | ■   | ■  |    |
| 6ESS 302-3....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| 307-3 interface module 1)                            |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     | 5)  | 5) |    |
| 6ESS 307-....  |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     | 4) | 4) |
| 308-3 interface module                               |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     | ■   | ■  |    |
| 6ESS 308-3....                                       |       |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |    |    |
| IP 240, 241, 242<br>243, 244                         |       |   |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| IP 245, 257, 260, 261                                |       |   |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| IP 246, 247, CP 513, 524,<br>525, 526, 527, 551, 552 |       |   |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| IP 242A, 2 etc., 252<br>CP530, 143, 580 1)           |       |   |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| I/O modules<br>DI/DQ/AI/AQ                           |       | ■ |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| HW alarm evaluation<br>of DI 432 and IPs             |       |   |    |    |    | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■  | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■   | ■  |    |
| PG-MUX slots with 6)<br>COOR 923C 3)                 |       |   | 1  |    | 1  |    |    | 2  |    | 2  |    |    | 3  | 4  | 5   | 6   | 7   | 8   |     |     |     |     |    |    |

- 1 CPU 946, CPU 928 and CPU 928B, AS 307, CP 535 and IM 307 take up two slots each
- 2 Jumpers 1 to 16 on the bus board which are accessible after removing the rear cover plate must be soldered in place.  
As delivered: slots 139/147 not suitable for these modules
- 3 Slots 91 to 131 are especially well suited for IPs and CPs since they have MUX capacity.
- 4 Here, no interrupt link to EU or ER is possible.
- 5 These slots are suitable for interrupt relaying according to 2.
- 6 The numbers in the table indicate the sub-addresses for PG communication via the PG multiplexer.

Table 2 Possible configurations for the S5-155U central controller

CPU 946/947 can only be operated as a unit; CPU 946 and CPU 947 cannot function individually.

|   |  |
|---|--|
|  | <p><b>Caution</b></p> <p>Do not plug modules into slots for which they were not intended as this can destroy these or other modules.</p> |
|---|--|

### 1.2.2 Possible Configurations for the S5-155H Programmable Controller

If you equip two 155 U central controllers each with one CPU 946R/947R and connect them together via the parallel connection IR 304/IM 324 R the resulting device is a central controller S5-155H. One 155U CC is then master and the other standby.

Multiprocessor operation is **not** possible with the S5-155H. The CPU 946R/947R can only be used once, whereby the slots 11, 19 and 27 are to be employed. The coordinator 923C may not be used.

The following table shows the possible configurations for the S5-155H:

| Slots                                    | 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 | 67 | 75 | 83 | 91 | 99 | 107 | 115 | 123 | 131 | 139 | 147 | 155 | 163 |  |
|--|---|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|--|
| CPU 946R 1)<br>6ES5 946-3UR..            |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| CPU 947R 1)<br>6ES5 947-3UR..            |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 355 memory module<br>6ES5 355-3UA..      |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| Parallel connection<br>IM 304/IM 324R 2) |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 301-3 interface module<br>6ES5 301-3.... |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 304-3 interface module<br>6ES5 304-3.... |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 300-5 interface module<br>6ES5 300-5.... |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 301-5 interface module<br>6ES5 301-5.... |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| 308-3 interface module<br>6ES5 308-3.... |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| I/O modules<br>DI/DQ/AI/AQ               |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| CP 524,525,526,527,551,<br>CP580         |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |
| CP 530 and CP 143                        |   |    |    |    |    |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |     |  |

- 1 CPU 946R takes up two slots
- 2 IM 304 in central controller B, IM 324R in central controller A (see Instructions for S5-155H)
- 3 If CPs are to be plugged in, jumpers 1 to 16 on the bus board which are accessible after removing the rear cover plate must be soldered in place.  
As delivered: slots 139/147 not suitable for CP.

Table 3 Possible configurations for S5-155H central controller

### 1.2.3 Addressing the I/O Modules

I/O modules can be addressed in the P and/or O (extended) I/O areas.

- I/O modules to be addressed in the O area must be plugged into the expansion unit. Depending on which combination of interface modules you require to communicate, you must set the address area either on the expansion unit interface module 300, 301, 307 or on the central controller interface module 314, 318.

It is also possible to multiply the O area:

- By setting "O area pages" on the IM 308 interface module a multiplexer function can be implemented which multiplies the O area by 256. The "O page number" must first be entered in O byte 255 before the operations L/T OB or L/T OW can be used to access the I/Os. The "O page numbers" are set on the EPROM of the IM 308. When using this procedure, the O area described above is not available.

#### Caution



#### To prevent double addressing:

If using an input module in the extended address area (O area) in an expansion unit (EU), make sure that there is no input module under the same I/O address in the central controller. The same applies for output modules.

### 1.3 Device Configuration of the S5-155U Programmable Controller

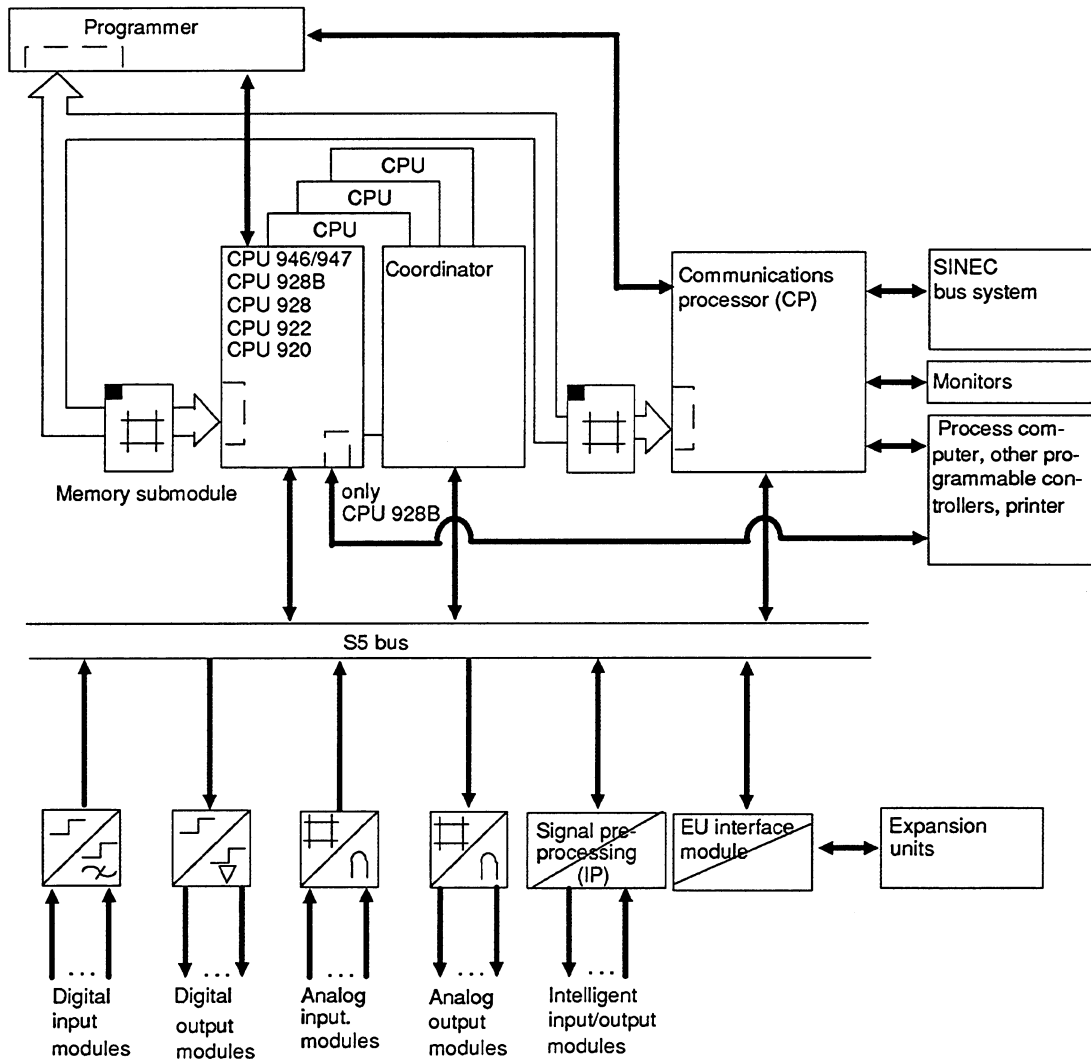


Fig. 2 Device configuration of the S5-155U programmable controller

### 1.4 Interfacing Between Central Controllers and Expansion Units

Expansion units can be connected if the number of slots in the central controller is insufficient. In this case refer to the Instructions of the expansion units (Manual U-Periphery).

When addressing the P or O I/O area note the following:

To avoid double addressing of the modules, do not use the same addresses in the I/O area (P area) of the central controller as in the extended I/O area (O area) of the expansion units. You can fully use the P and O I/O area if all I/O modules are in the expansion units. With the S5-155H programmable controller systems refer to the S5-155H Instructions.

### 1.4.1 Central Interfacing

Central interfacing means that the expansion units are accommodated together with the central controller in the same cabinet or in an adjacent cabinet. The total cable length from the central controller to the furthest expansion unit must not exceed 2 m.

The example (Fig. 3) shows how the S5-155U CC and the EG-183U/184U expansion unit are connected with the appropriate interface modules.

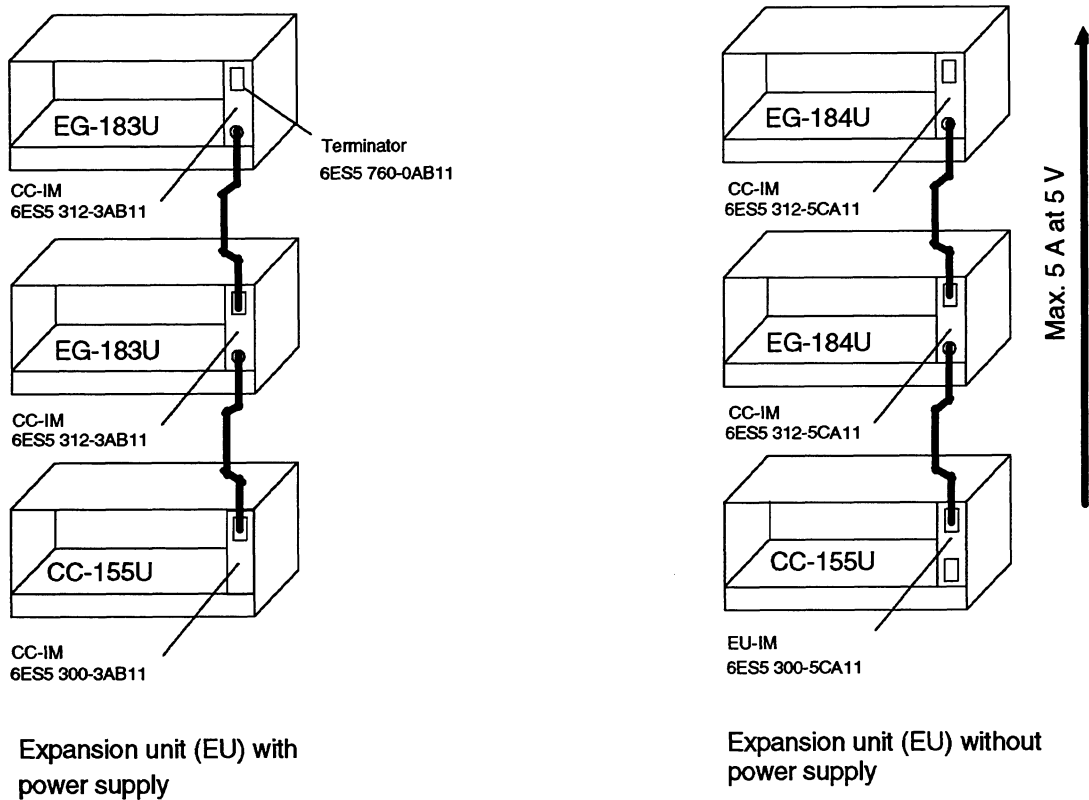


Fig. 3 Example of central design with S5-155U central controllers and EG-183U / S5-184U expansion units.

Please note:

- A terminator must be used on the last CC-IM 312-3....
- If you use expansion units without a power supply, the maximum load on the interface cable is 5 A (at 5 V).

For further designs with other interface modules, see Catalogs ST 54.1/ST 52.3 and the manual "U-Periphery" 6ES5 998-0PC22.

## 1.4.2 Distributed Interfacing

Distributed interfacing means that the expansion units are accommodated in a cabinet located further away from the central controller. The total cable length from the CC to the most remote EU must not exceed defined values. These distances depend on the interface module used (see Section 1.4.3, table: Examples of further interfacing possibilities). A distributed configuration up to 600 m is shown in the example (Fig. 4).

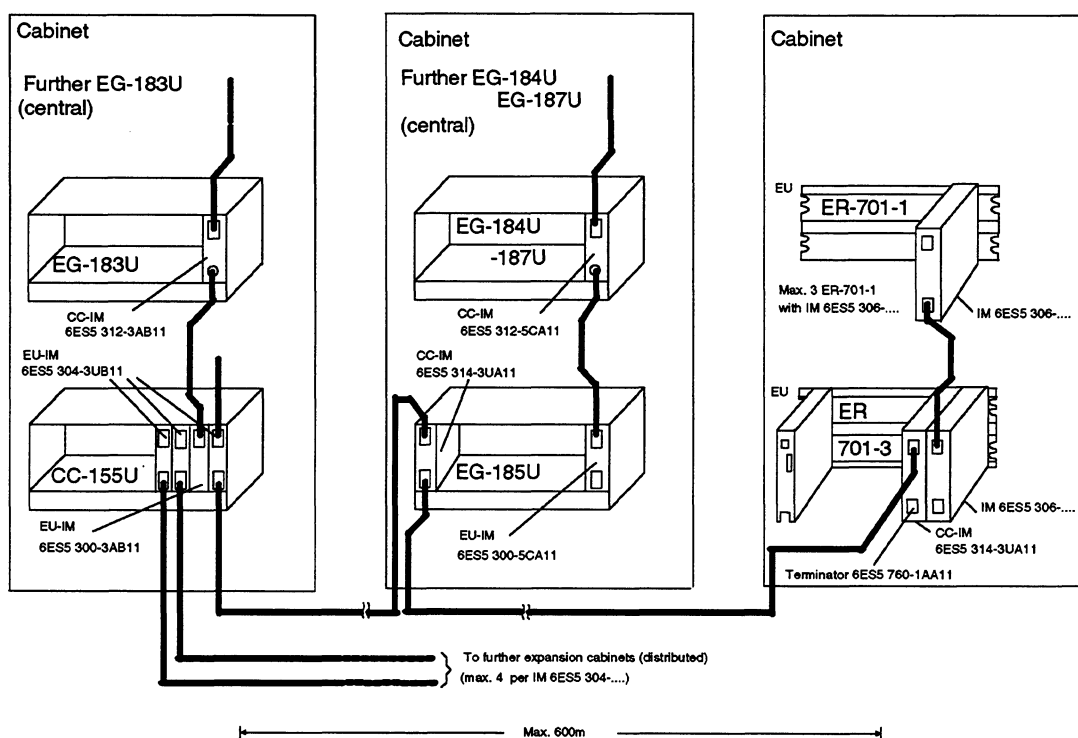


Fig. 4 Example of a distributed configuration up to max. 600 m with S5-155U central controller and EG-183U/184U /187U expansion units, ER 701-1 / 701-3 subracks

Please note:

- A distributed connection of up to 2 lines with 4 subracks (EG-183U, EG-185U, EG-186U expansion units, ER 701-2 and ER 701-3 subracks) can be made to a CC using an EU-IM 304 via the CC-IM 314.
- The total length (cable connector 721) from the CC up to the last EU can be up to 600 m per line.
- Further subracks can be connected centrally to the EG-185U expansion units and ER 701-3 subracks connected with a distributed configuration.
- Terminators must be inserted in the last CC-IM 314 of each line and in the last central connection of the CC-IM 312-3.



For further designs with other interface modules, see Catalogs ST 54.1/ST 52.3 and the manual "U-Periphery" 6ES5 998-0PC22.



**Caution:**

Only original cable connectors must be used to connect the interface modules to one another.

The screen of these cables is connected at both ends (do not isolate!).

The screen connection to the CC/EU subrack must be guaranteed via the springs on the metal front panel of the interface module or - in the case of plastic front panels - via the springs in the guide rails of the subrack.

Ensure that these important contact springs are not bent or dirty and are not interrupted at any point.

### 1.4.3 Examples of Interfacing Possibilities

The following table shows which interface modules and cable connectors can be used to connect the various expansion units to the central controller.

For S5-155H systems note the S5-155H Instructions

| Type of design           | Interface module in central controller | Expansion unit  | Interface module in expansion unit | Cable connector                       |
|--------------------------|--|---|------------------------------------|---------------------------------------|
| Central                  | 6ES5 300-3AB11<br>6ES5 301-3AB13       | EG 183U   | 6ES5 312-3AB11<br>6ES5 312-3AB31   | Fixed on module 312<br>"              |
|                          | 6ES5 300-5CA11<br>6ES5 301-5CA12       | EG 184U<br>EG 187U  | 6ES5 312-5CA11<br>6ES5 312-5CA21   | "<br>"                                |
|                          | 6ES5 300-5LB11                         | ER 701-1  | 6ES5 306-7LA11                     | 6ES5 705-0xxxx                        |
| Distributed up to 200 m  | 6ES5 301-3AB13                         | ER 701-2  | 6ES5 310-3AB11                     | 6ES5 721-0xxxx                        |
|                          | 6ES5 301-5CA12<br>6ES5 301-3AB13       | EG 183U   |                                    |                                       |
| Distributed up to 600 m  | 6ES5 304-3UB11                         | ER 701-2<br>ER 701-3<br><br>EG 183U<br>EG 185U<br>EG 186U | 6ES5 314-3UA11                     | 6ES5 721-0xxxx                        |
| Distributed up to 3000 m | 6ES5 308-3UA12                         | ER 701-2<br>ER 701-3<br><br>EG 183U<br>EG 185U<br>EG 186U | 6ES5 318-3UA11                     | Screened, twisted 2-wire cable        |
|                          |  | ET 100U<br><small>(Catalog ST 52.1)</small>               | 6ES5 318-8MA12                     |                                       |
|                          |  | ICM 560   | -                                  |                                       |
| Distributed up to 1500 m | 6ES5 307-3UA11                         | ER701-2<br>ER701-3<br><br>EG183U<br>EG185U<br>EG186U      | 6ES5 317-3UA11                     | 6ES5 722-2xxxx<br>(fiber-optic cable) |

Table 4 Examples of interfacing possibilities



**Note:**

An equipotential bonding conductor  $\geq 10 \text{ mm}^2$  must be provided for a distributed connection of central controllers and expansion units via the interface module 301/310 (up to 200) or 304/314 (up to 600) if a potential difference of more than 7 V can occur (see IEC 364-4-41, VDE 0100). The interface modules 301/310 and 304/314 are non-floating.

Connection via fibre-optic interface module IM 307/317:

The fibre-optic interface module IM 307/317 provides an isolated connection that can also transfer interrupts from an expansion unit (EG 186U or ER 701-3) to a central controller (on the slots 139 and 147, provided that the jumpers 1 to 16 on the rear bus PCB are closed).

## 1.5 Mode of Operation

The S5-155U programmable controller belongs to the SIMATIC S5 range. The controller can be used as a single processor as well as for multiprocessing with up to four CPUs. In multiprocessing operation a 923C coordinator is required. The exception is the CPU 946R/947R for S5-155H, which cannot be used for multiprocessing.

### 1.5.1 Single/Multiprocessor Operation (only for S5-155U Programmable Controller)

The user program is executed cyclically. Access to the input and output modules is possible at all times via the S5 bus. If the 923C coordinator is present, you can configure the S5-155U programmable controller with more than one CPU.

The S5-155U programmable controller is a multiprocessing device with processors for specific tasks which can be combined in a variety of ways:

- CPU 946/947      Fast processing of all STEP 5 operations. With a maximum of 896 kbytes this CPU offers maximum memory configuration.
- CPU 928B        Designed for multiple tasks; provides very fast binary signal processing (open-loop control tasks) as well as very fast word processing (computing and closed-loop control). Has a second interface for point-to-point link with external devices (PC, PLC, PG, printer...).
- CPU 928         Designed for multiple tasks; provides fast binary signal processing (open-loop control tasks) as well as fast word processing (computing and closed-loop control).
- CPU 922         Mainly for fast word processing (computing and closed-loop control).  
(R processor)      Binary signal processing is also possible.
- CPU 920         For processing of measured values, arithmetic and statistics.  
(M Processor)      Programming is carried out in BASIC, C or Assembler.

The automation task can be clearly organized by using several CPUs. Each CPU executes its program independent of the others. This increases the overall processing speed. Each processor can be started independent of the others. Up to four processors can be operated in the programmable controller, and the user program can be divided amongst several CPUs for specific tasks. All data to be exchanged between CPUs is transferred via the S5 bus.

In multiprocessor operation, you must specify address lists in data block DB 1 of each CPU to allow the input/output modules to be assigned to the individual CPUs. DB 1 must not be programmed as a standard DB. In single processor operation, DB 1 can be used to increase the processing speed.

The coordinator assigns each processor access to the S5 bus cyclically. Information is exchanged between the processors via the coordinator, which has a memory for this purpose. The coordinator 923C has an interprocessor communication flag area, a memory for multiprocessor communication and a central programmer connection. Via this, up to 8 modules (CPU, CP, IP) can be accessed without reconnecting the cable to the PG or SINEC-CP.

## **1.5.2 Unit Interfacing**

The S5-155U central controller can be connected to:

- **Programmers**

The programmers can be directly connected to the processors or the 923C coordinator for programming or system start-up.

PG multiplexer function: with an electronic switch on the 923C coordinator you can access up to 8 modules in the central controller from the PG.

- **Standard and peripheral devices and computers**

The communications processors (CP) can handle data traffic independently with

- standard peripheral devices such as printers, keyboards, monitors,
- computers or
- other programmable controllers.

The data required for texts and displays can be programmed for each communications processor in a RAM or EPROM submodule.

The CPU 928B can handle data traffic independently via its second interface with

- standard peripheral devices such as printers, keyboards,
- computers or
- other programmable controllers.

- **SINEC buses**

The communications processors (CP) can handle data traffic independently with

- PGs, computers or
- other programmable controllers.

You can also use CPs for operation and monitoring, to display and/or modify process data. CPs are also available to display diagnostic messages and there are also CPs with mass memories.

## 2 Installation of the S5-155U Central Controller

### 2.1 Installing the Central Controller

The S5-155U central controller is designed for installation in cabinets, on frameworks and on walls (see Section 3.3 in Part 2; Specifications when Installing a Cabinet and Section 2.7; Framework and Wall Mounting of SIMATIC S5 Controllers).

The S5-155U central controller need only be accessible from the front for carrying out connections and maintenance.

The following Figs. show you the dimensions relevant to installation of the central controller.

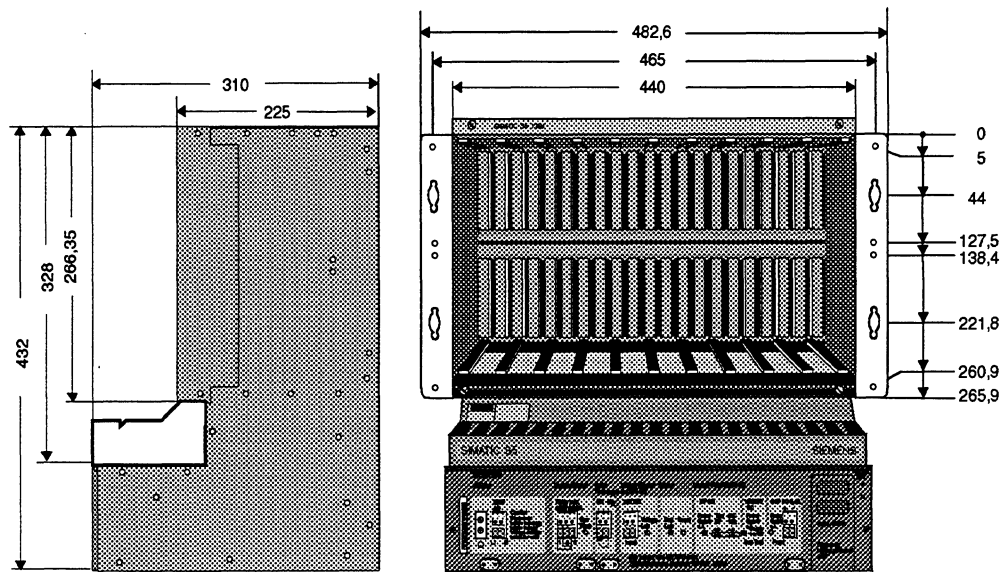


Fig. 5 Device dimensions of the S5-155U central controller (in mm)

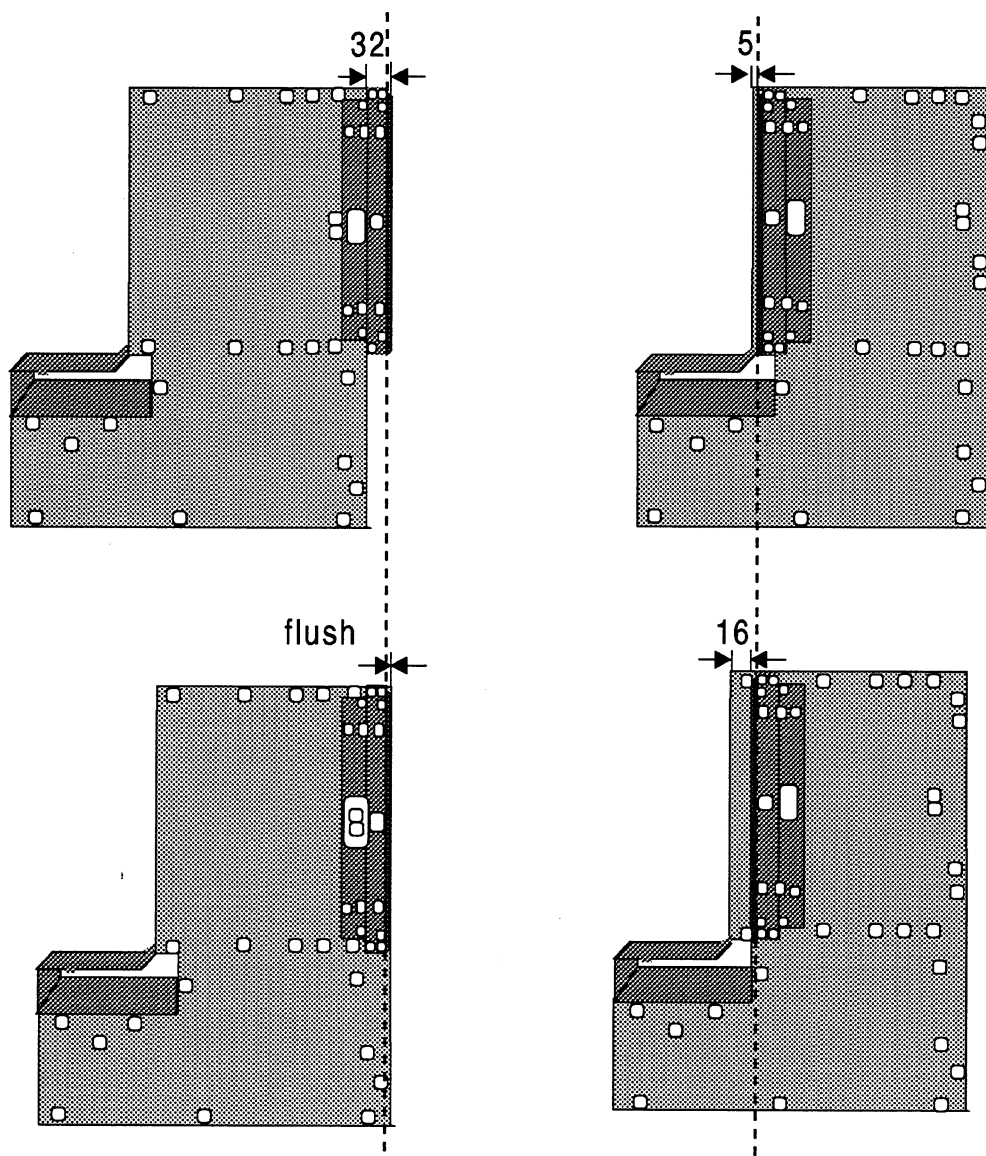


Fig. 6 Different ways of fixing mounting brackets

**Note :**

➔ The same installation dimensions apply to the EG-183U/184U/185U and 186U expansion units as to the central controller.

## 2.2 Installing the Power Supply Unit

Insert or remove the power supply unit only when the device is switched off.

Push the power supply unit to the back. Push it firmly against the limit stop until the panel is flush with the device frame. The spring action of the contacts must be overcome. Then tighten both screws in the frame at the right and left of the front panel. The protective jumper at the left must be securely connected to the front panel terminal and the rack of the central controller.

### 2.2.1 Installing the 15-V Supplementary Module

The supplementary module <sup>1)</sup> must only be inserted when no voltage is applied.

Remove the power supply unit as described in Sections 2.2 and 5.2, and insert the 15-V supplementary module in the space shown in Fig. 7.

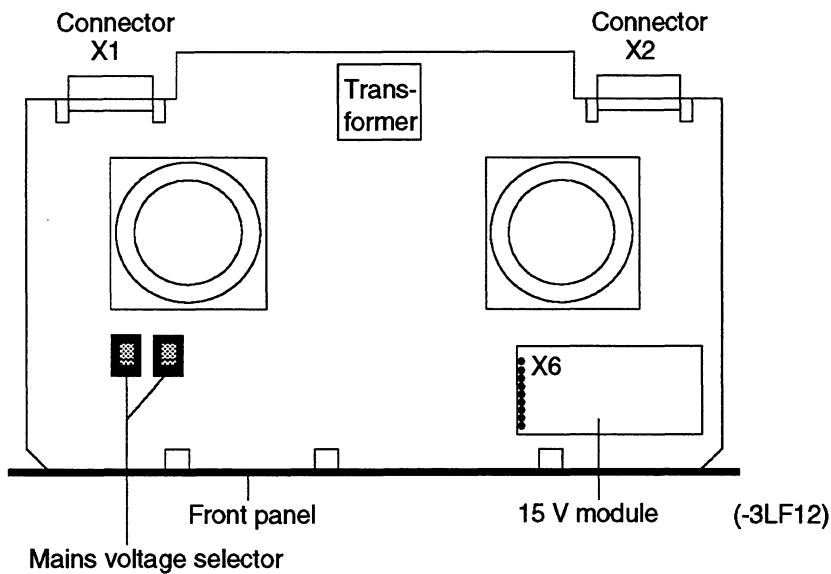


Fig. 7 Mounting location of the 15-V supplementary module

<sup>1)</sup>See "Ordering Information" for Order No. in the Appendix

## 2.3 Installation of the Modules

The dimensions of the S5-155U programmable controller modules correspond to the double-height Europa format (w x h x d: 20.32 mm x 233.4 mm x 160 mm).

There are modules with different widths:

| Slots occupied | Standard slots <sup>1)</sup> | Front panel width | Module                             |
|----------------|------------------------------|-------------------|------------------------------------|
| 1              | 1 <sup>1/3</sup>             | 20.32 mm          | e.g. CPU 947,<br>memory module 355 |
| 2              | 2 <sup>2/3</sup>             | 40.64 mm          | e.g. CPU 946,<br>CPU 928B          |
| 4              | 5 <sup>1/3</sup>             | 81.28 mm          | e.g. CP 580                        |

1) Standard slot in ES 902 = 15.24

Table 5 Examples of widths of modules

Observe the following when installing the modules:

- Insert each module into the subrack as far as possible
- Lock the modules with individual locking mechanisms
- Screw the top locking rail for modules onto the central controller.

To improve the ventilation and the degree of protection, you can cover vacant slots using dummy front panels. See "Ordering Information" in the Appendix.

### 2.3.1 Connections to the CPUs, CPs and Interface Modules

Connect the cables from CPUs, communications processors and expansion unit interface modules using front plugs. There are two kinds of metal front plugs:

1. Metal front plugs with a sliding locking mechanism are locked by pushing the sliding bracket down.
2. Metal front plugs with a thumb wheel screw are screwed to the device.

Take care to assign the plugs to the correct modules as damage could otherwise result.

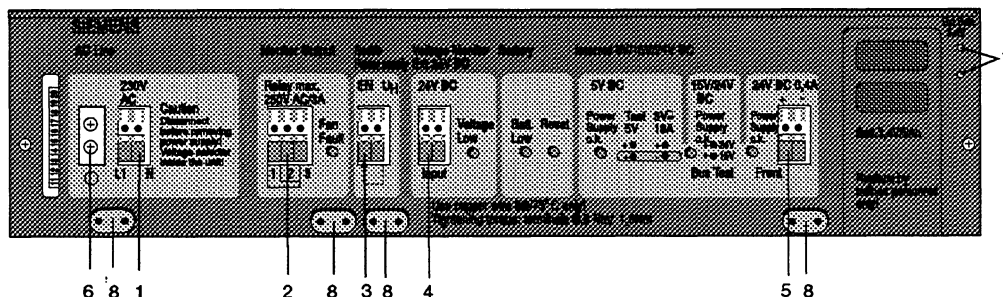
### 2.3.2 Connections to the I/O Modules

Information on how to connect the cables to the I/O modules can be found in the U-Periphery Manual (Order No. 6ES5 998-0PC...) and in the ST catalogs.



### 3 Wiring Connections on the Power Supply Unit of the S5-155U Central Controller

#### 3.1 Connections of the Power Supply Units



- 1 AC line :  
230 V/120 V selectable input voltage (a 24 V DC line is also possible depending on the type of power supply unit).
- 2 Monitor output :  
external signalling on LED and relay contact if one or both fans stops; this results in the output voltages being switched off (function selectable using jumper F-R of the power supply unit; in this case only relay signal and LED display). In addition, failure of the back-up battery can be signalled on the power supply unit with the jumper RR-LL closed.  
- See Section 4.3. for the functions and locations of the jumpers on the power supply units.  
- See Section 3.2. for recommended wiring.
- 3 Enable power supply :  
the power supply unit is switched off if no voltage is present at the EN input. Not more than 7 EN inputs (front terminal) can be set with an U<sub>H</sub> output.  
- See Section 3.2. for recommended wiring.
- 4 Voltage monitor :  
24 V load voltage monitor input, must be connected or switched inactive by means of jumper BA-EX in the power supply unit. This does not apply to the power supply unit 6ES5 955-3NA12.
- 5 24 V DC, 0.4 A output:  
this output can be used to supply the enable inputs of the U periphery.
- 6 Protective ground conductor connection:
- 7 Connection socket for external back-up voltage of 3.4 to 3.6 V.
- 8 Cable detensioners for connection cables with metal contact surface for cable screens.

Fig. 8 Connections of the power supply unit



**Caution :**

The appropriate safety regulations must be adhered to, especially IEC 364-4-41. The terminals at the front are suitable for cables with a cross-section up to 4 mm<sup>2</sup>. The 230/120 V AC power connection must have a core cross-section of at least 0.75 mm<sup>2</sup>. Ensure that the tension on the cables is relieved sufficiently.

### 3.2 Recommended Wiring for Fans and Temperature Monitoring

If several devices with fan subassemblies are to be monitored together, connect the terminals EN and U<sub>H</sub> of the power supply unit according to the following diagram. **All devices are switched off if the fan fails in one device.** The jumper settings required on the power supply units for this purpose are described in the following sections.

If you have installed the device in a cabinet with heat exchanger, the maximum internal temperature could be exceeded even with the fans working.

It is advisable to install a temperature monitoring device or thermostat in the top of the cabinet (in the exhaust air flow) (see Catalog NV 21). If a fault develops, the signal from the fan or temperature monitoring should be transferred immediately to the CPU to activate a programmed response. The actual shut-down should be triggered by a (customer-installed) time-delay relay after a maximum of 60 seconds, by interrupting the U<sub>H</sub>-EN loop.

Up to 7 EN inputs can be controlled using on U<sub>H</sub> output.

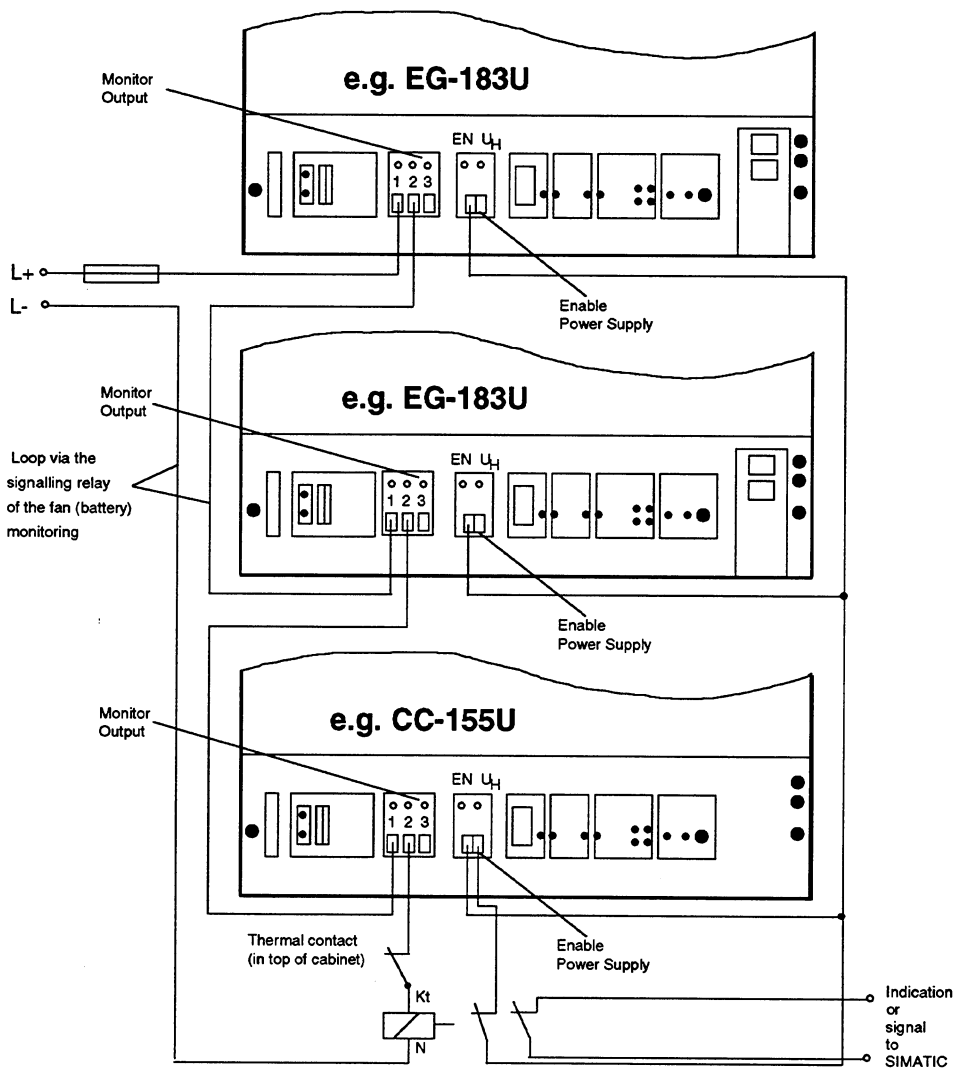


Fig. 9 Recommended connections for fan/temperature monitoring when using S5-155U CC and EG-183U EUs



**Caution**

Modules with a Winchester disk drive can only be used at an ambient temperature of up to 50 °C [122 °F]. The permissible switching point of the thermal contact in the top of the cabinet must be adapted. If necessary, use an adjustable thermostat (see Catalog NV 21).

### 3.2.1 Setting the Fan Monitoring

You can use the jumper F-R on the power supply units to select whether the internal supply voltages  $U_A$  (5 V) are to be switched off or not in the event of a fan failure:

- Jumper F-R closed:  $U_A$  switched off (signal via contact)
- Jumper F-R open:  $U_A$  not switched off (signal via contact).

The signalling relay ("Monitor Output") is activated if one or both fans stops. The LED "Fan Fault" lights up at the same time.

- Relay contact 2-1 closed: fan running
- Relay contact 2-3 closed: fan failure

Relay contact 2-3 closed is also the normal position, i.e. position when the power is off (intrinsic safety). The position of the jumpers on the power supply unit is shown in Section 4.3.



**Caution**

Jumper F-R must be opened if switching-off cannot take place immediately. In this case you must ensure that the power supply is switched off at the latest after 60 s. This can be achieved e.g. using a time-delay relay. This prevents modules being damaged by overheating.

### 3.2.2 Setting the Back-Up Battery Monitoring

Using jumper RR-LL on the power supply unit you can select whether the signalling relay ("Monitor Output") is to be switched not only upon a fan failure but also upon a battery failure:

- Jumper RR-LL open (factory setting): relay only signals fan failure.
- Jumper RR-LL closed: relay signals fan and battery failures.

The signalling relay ("Monitor Output") is switched in the event of a battery failure or standstill of one or both fans. The LED "Batt Low" lights up at the same time.

- Relay contact 2-1 closed: battery OK or (optionally) fan running.
- Relay contact 2-3 closed: battery faulty or (optionally) fan failure.

The position of the jumper on the power supply unit is shown in Section 4.3.

#### Note



The signalling relay is activated if a fan or the battery fails. The signalling relay must be wired by the user to adapt it to both faults. If the signalling relay is triggered in the event of a battery failure, and as a result the PLC is switched off, the program in the user memory is lost. This can be avoided by having an external back-up voltage applied to the sockets on the front panel of the power supply unit while the PLC is being switched off (see Fig. 8).

## 4 Operation of the S5-155U Central Controller

Before you start up the programmable controller, read the notes in the following section. These notes explain the requirements for operating a PLC and contain useful information about starting up and operating the S5-155U system.

The S5-155U programmable controller of the type 6ES5 155-3UA1x is set at the factory for operation with 230 V AC. If you want to operate the PLC with 120 V AC remove the power supply unit and change the slider switch settings to 120 V (see Fig. 7). Before refitting the unit please stick the label "120 V" over the printed label "230 V". When you refit the power supply unit make sure that the short protective ground cable is reconnected to the power supply unit and housing.

The S5-155U programmable controller of the type 6ES5 155-3UA2x is designed for an operating voltage of 24 V DC.

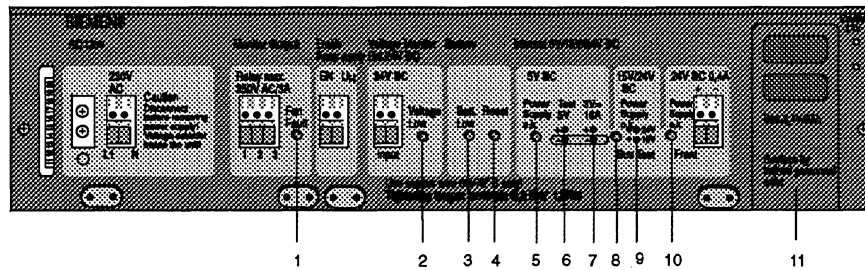
### 4.1 General Notes on the Power Supply Unit

- No voltages > 50 V must occur between the power supply outputs and the protective ground conductor of the power supply unit.
- The protective ground conductor must always be connected as well as the jumper between the CC rack and the front panel of the power supply unit.
- If there is an overvoltage at the internal DC supply voltages  $U_{01} = +5 \text{ V}$  and  $U_{03} = +15 \text{ V}$ , the power supply unit is switched off without loss of data. A voltage  $\leq 0.5 \text{ V}$  (see Chapter 6, Technical Data, for overvoltage switch-off) is present at  $U_{01}$  and  $U_{03}$  in the switched-off condition.  
The power supply unit can be started up again by switching the external power supply off and then on again to reset the memory flip-flop, provided that the overvoltage was not caused by an internal fault.
- The power supply unit will only function correctly if the +5 V side has a load of at least 1 A (2 A with power supply unit -3LF12).
- An air filter <sup>1)</sup> can be installed in the base of the power supply unit housing.
- Make sure the voltage level (3.4 V) and polarity are correct when using an external back-up voltage.
- **The back-up battery is supplied loose and must be fitted before you start up. Without the battery, the PLC will not start when the power is switched on. The battery must first be inserted. Then press the RESET button. Then perform an overall reset.**
- The wire jumper for "enable power supply" from  $U_H$  and EN enables the power supply. By suitable wiring of the monitoring outputs with the EN inputs, you can block the PLC in the event of a fault (see Section 3.2).

1) See Catalog ST 54.1 and Appendix for order nos.

- Undervoltage or the absence of voltage at the input terminals "voltage monitor" for 24 V DC activates the signal BASP in the CC and connected expansion units so that all digital outputs are disabled (the function can be disabled with jumper BA-EX). In this situation the CPUs do not recognize that the BASP signal is active. If you want the CPUs to recognize this, a digital output must be set constantly to 1 and be connected to an input that is scanned at least once per cycle.

## 4.2 Operating and Display Elements of the Power Supply Unit



- 1 LED "Fan Fault"  
The red LED lights up if a fan fault has occurred. The power supply unit then switches off (jumper F-R closed) with a delay of approx. 6 to 10 s. The jumper F-R must be opened if the programmable controller cannot be switched off immediately for technical reasons. The programmable controller must always be switched off within 60s (overheating of module).
- 2 LED "Voltage Low"  
The red LED lights up if an undervoltage is present at the load voltage monitor input (omitted with power supply unit -3NA12).
- 3 LED "Batt. Low"  
The yellow LED lights up if the battery voltage has dropped below 2.7 V; the data buffered in the RAM are lost following "power off/on".
- 4 Key "Reset"  
If the programmable controller is in the power-off state, the battery must be replaced following mains-on and with the LED "Batt. Low" lit. Press this key after replacing the battery, the LED "Batt. Low" then goes out.
- 5 LED "Power Supply O.K."  
The green LED lights up when the output voltage of 5 V is present.
- 6 Test sockets "Test 5 V"  
For checking the output voltage  $V_{o1}$  (standard setting: 5.1 V DC  $\pm$  0.5 %)
- 7 Test sockets "3 V  $\hat{=}$  18 A or 40 A"  
For checking the output current  $I_{o1}$  (3 V  $\hat{=}$  max. output current of respective power supply unit).
- 8 LED "Power Supply O.K." (Bus)  
The green LED lights up when the output voltage of 15 V (if the 15 V supplementary module is used) **and** the output voltage of 24 V are present.
- 9 Test sockets "15 V/24 V DC" (Bus)
  - a) For checking the output voltage  $V_{o2}$  (24 V DC +25 %/-17 %)
  - b) For checking the output voltage  $V_{o3}$  (15 V DC  $\pm$ 5 % provided the 15 V supplementary module is plugged in)
- 10 LED "Power Supply O.K." (terminal)  
The green LED lights up when the output voltage for the enable supply is present at the terminal "24 V DC".
- 11 Battery plug-in

Fig. 10 Operating and display elements of the power supply units

### 4.3 Functions and Locations of Jumpers on the Power Supply Unit 6ES5 955-3xyy

| Function   | Jumpers   |
|--|---|
| Battery monitoring ( $\overline{\text{BAU}}$ ) ON<br>Battery monitoring (BAU) OFF  | NN-MM closed *<br>NN-MM open  |
| Power supply unit switched off after fan fault<br>Power supply unit not switched off after fan fault<br>(only signal LED and relay)  | <b>F-R closed *</b><br>F-R open   |
| Operation with load voltage monitoring<br>Operation without load voltage monitoring  | <b>BA-EX open *</b><br>BA-EX closed   |
| Activation of signal relay in the event of a fault<br>(relay contact 2-3 closed) <ul style="list-style-type: none"> <li>• <b>by fan fault</b></li> <li>• <b>by battery fault signal <sup>1)</sup></b></li> <li>• <b>without battery fault signal</b></li> <li>• <b>by low voltage signal <sup>2)</sup></b><br/>(BASPA = low)</li> <li>• <b>without low voltage signal</b></li> </ul> | <p>Independent of other jumpers</p> <p>RR-LL closed</p> <p><b>RR-LL open *</b></p> <p>BB-AA closed</p> <p><b>BB-AA open *</b></p> |

- 1) Battery undervoltage ( $V_{\text{BATT}} < 2.7 \text{ V}$ ), leads to a battery fault signal (can be switched off using jumper MM-NN). In addition to display "Batt. Low" and output of the signal BAU, the signal relay can be activated via the jumper RR-LL if the following power supply units are used:

Power supply units:

6ES5 955-3NF11 from version 7 onwards  
6ES5 955-3LF12 from version 5 onwards

Jumper RR-LL does not function in other power supply units.

- 2) If there is an undervoltage at the monitor input ( $V_M < 20 \text{ V}$ , -25 % can be switched off via jumper BA-EX) or at the 5V output ( $V_o < 4.75 \text{ V}$ ), the "BASPA = low" signal is output. Thus the indirect and digital outputs are inhibited.

Table 6 6ES5 955-3LF12 power supply unit jumper assignment

\*) Factory setting



There is a 6 A fast-acting input fuse at position F107 and a 4 A fast-acting output fuse at position F255. For the 6ES5 955-3NF11 power supply unit, the input fuse is at position F110 and the output fuse is at position F117. The position and the number are printed on the PCB.

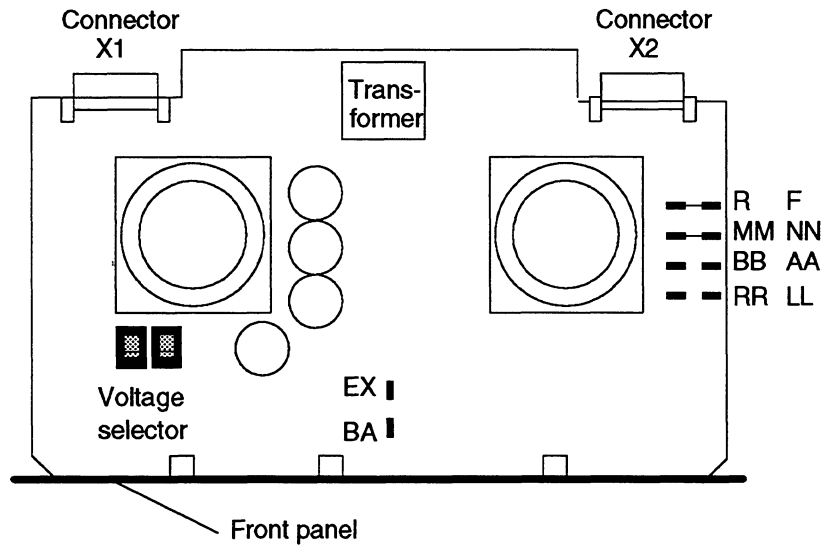


Fig. 11 Power supply unit 6ES5 955-3LF12

#### 4.4 Power Supply Behaviour in the Event of Faults

If the power supply is switched off, relay contact 2-3 is closed and relay contact 1-2 open.  
 In normal operation, relay contact 1-2 is closed and relay contact 2-3 open.  
 In addition to fan faults, other faults (see jumper description) can set the signal relay to the normal position (relay contact 2-3 closed) by means of appropriate jumper settings.

The following table shows the response of the power supply in the event of faults (condition: jumper MM-NN closed, jumper BA-EX open).

| Fault and jumper settings     |              |          |  | Reaction     | Message         |                      | Output voltage $V_o$ (5V) switched off |               |
|-------------------------------|--------------|----------|--|--------------|-----------------|----------------------|--|---------------|
|                               |              |          |  |              | LED "Fan fault" | Closed relay contact |  |               |
| Enable (jumper EN-UH) present | No fan fault |          | No battery fault and no undervoltage message |              | Dark            | 1-2                  | No                                     |               |
|                               |              |          |  |              |                 |                      |  | Battery fault |
|                               |              |          | RR-LL open                                   | 1-2          |                 |                      |  |               |
|                               |              |          | Undervoltage message                         | BB-AA closed |                 |                      |  | 2-3           |
|                               | BB-AA open   | 1-2      |  |              |                 |                      |  |               |
|                               | Fan fault    | F-R open |  | Lights up    |                 | Yes                  |  |               |
| F-R closed                    |              |          |  |              |                 |                      |  |               |
| No enable (jumper EN-UH)      | No fan fault |          | No battery fault                             |              | Dark            |                      |  |               |
|                               |              |          |  |              |                 |                      | Battery fault                          | RR-LL closed  |
|                               |              |          | RR-LL open                                   |              |                 |                      |  |               |
|                               |              |          | Undervoltage message                         | BB-AA closed |                 |                      |  |               |
|                               | BB-AA open   |          |  |              |                 |                      |  |               |

Table 7 Fault message and reaction of the power supply unit

### 4.5 Start-Up and Functional Test (only for S5-155U Programmable Controller)

Requirement: 1 S5-155U with one processor (CPU)

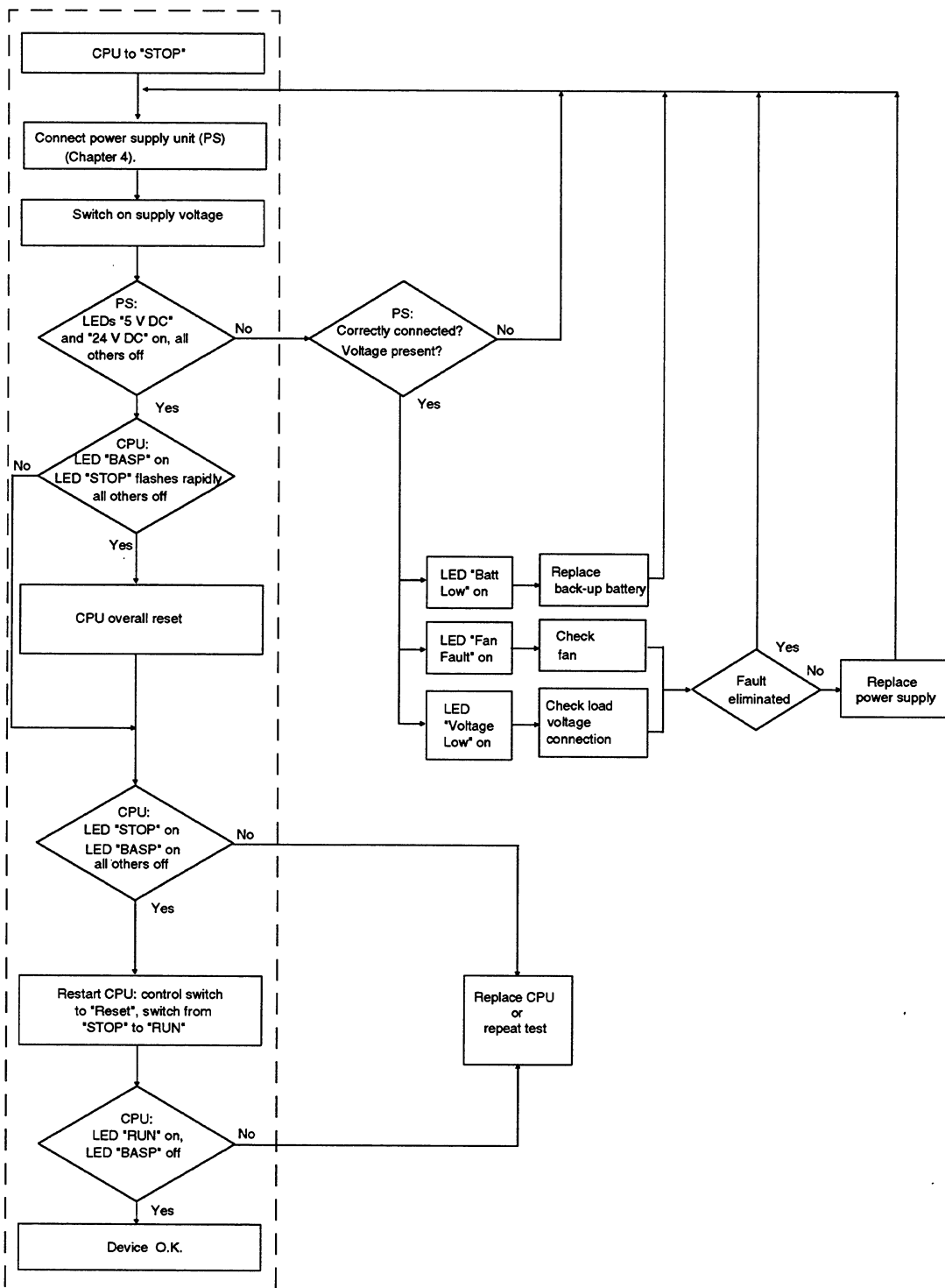
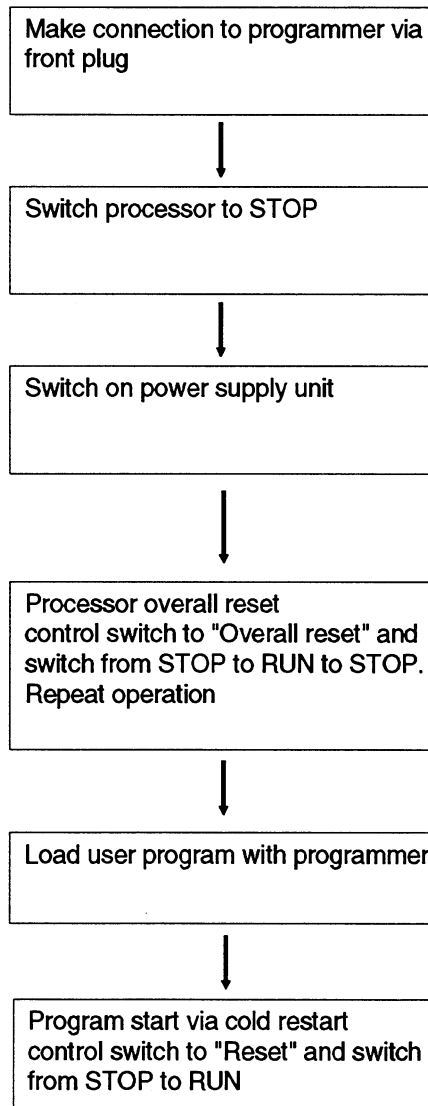
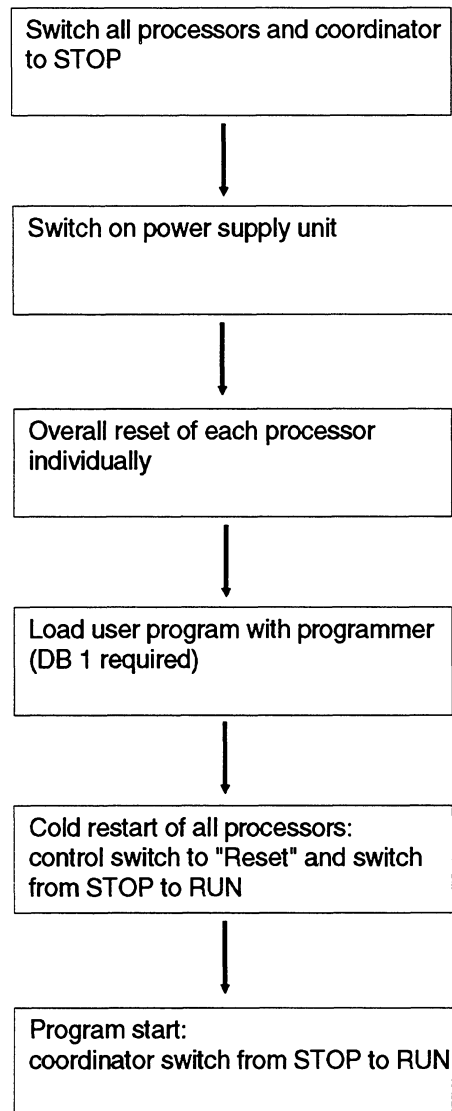


Fig. 12 Starting up

#### 4.5.1 Start-Up with Single Processor Operation



#### 4.5.2 Start-Up with Multiprocessor Operation (only for S5-155U Programmable Controller)



### 4.5.3 Checklist for Starting Up

Start up the PLC in the order described here. This brings you to the first trial run with the CPU.

The chapters of the publications describing the steps in detail are shown in brackets.

To avoid the start up being too complex at this stage, you should simply begin with one CPU and without any expansion units.

- 1) Install the PLC so that air can circulate freely. If you are using several devices (PLC and EU) in one cabinet, make sure the required clearances are maintained and if necessary install air baffles (Installation Guidelines, Section 3.3).
- 2) Fit the back-up battery (Section 5.3).
- 3) Install the CPU and set the mode selector to STOP.
- 4) Connect the power supply to the monitoring input and when using the 230/120 V AC supply the 24 V load voltage.
- 5) Switch on the power and, if present, the 24 V supply: green LED "Power supply o.k." in the "5 V DC" and "15/24 V DC" fields and the yellow LED "Batt. Low" light up.
- 6) Press the RESET button on the power supply unit. The yellow LED "Batt. Low" goes out (Section 4.2).
- 7) OVERALL RESET of the CPU :  
Hold the CPU button in the "overall reset" position and switch the mode selector from STOP to RUN: the "STOP" LED flashes quickly.  
Repeat this step: the LED "STOP" is lit constantly.  
Hold the button in the "RESET" position and switch the selector from STOP to RUN: the green LED "RUN" lights up, the small yellow LED "BASP" goes out (Section 4.5).

The CPU is now running through an empty cycle. After switching off the power supply, you can now plug the other modules into the rack and connect the expansion units. Remember to plug in modules only in the permitted slots (Section 1.2.1), the addressing possibilities with the P and O areas and the settings on the interface modules (U Periphery manual) and the installation guidelines (Part 2).

When starting up with more than one CPU remember that the address list must be programmed on DB 1 of all CPUs. Refer to the "Multiprocessor Operation Instructions" for detailed information about multiprocessing.

## 5 Maintenance of the S5-155U Central Controller / Pin Assignments

If measurements or tests are required on active devices, the accident prevention regulations (VBG 4.0) must be observed and suitable tools used



### Warning:

Repairs on automation equipment must only be carried out by the **Siemens servicing department** or by **qualified personnel** (see above).

Always remove the mains plug or open the isolating switch before opening the device.

Only use replacement fuses of the same type.

### 5.1 Removing and Inserting S5 Modules



### Warning:

I/O modules with an active enable circuit may only be removed during operation if the enable voltage is switched off. This is achieved when the front plug is removed.

All inputs of this module are written into the process image of the inputs as "zero" if the enable voltage is missing, if the front plug is disconnected or if the module is removed. With a direct access to the process I/Os, the inputs are also written into ACCU 1 as "zero".

With all the other modules you must first switch off the device before removing or inserting a module. Removing or inserting these modules while voltage is applied can damage the module and lead to undefined system statuses.

You must ensure in the organization block in which the acknowledgement delay is checked that a hazardous condition in the process or on the machine cannot occur if a fault occurs or when a module is replaced.



**Caution**

Dangerous voltages may be present at the front sockets of I/O modules 435, 436, 455 and 456 if the front plugs are removed and inserted during operation. When under voltage, modules must only be replaced by electricians or trained personnel.

If it is not necessary to remove and insert modules during operation, the wiring of the enable circuit (F<sub>+</sub>/F<sub>-</sub>) can be omitted on the I/O modules. The jumper for switching over the enable mode must then be removed.

After removing the jumper for the enable mode, the module can be addressed via the S5 I/O bus irrespective of how the enable circuit F<sub>+</sub>/F<sub>-</sub> is connected. Connection of the enable voltage is no longer necessary. You must never remove or insert I/O modules connected to voltage if the enable circuit is not activated since this can damage the module and lead to undefined system statuses.

## 5.2 Removing and Inserting Power Supply Units

Power supply units must only be removed when no voltage is applied.

The connection between the back-up battery and the backplane bus is retained when the power supply unit is removed, thus ensuring that the user program is still backed-up.



### 5.3 Replacing the Back-Up Battery and the Fans

The back-up battery can be changed without losing any data in the memory if the power supply unit is switched on or if an external voltage (3.4 V) is applied to the sockets "Ext. Batt.". The back-up battery should be replaced every three years regardless of the memory configuration or the extent to which it had been used.

Proceed as follows to replace the battery:

- Pull down the cover.
- Pull the battery module to the front and remove it.
- Replace the battery.
- Make sure the polarity is correct.
- Once the new battery is fitted and the power is on, press the RESET button on the power supply module (see Fig. 11).

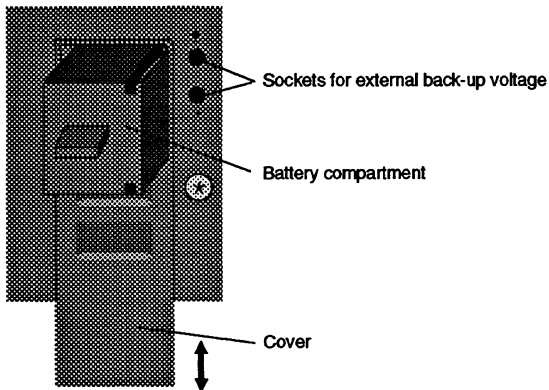


Fig. 13 Battery compartment



**Caution:**

Ensure correct polarity when fitting a battery or applying an external back-up voltage.



**Caution: LITHIUM THIONYL CHLORIDE BATTERY!**

Do not dispose of batteries in fire and do not solder on cell body - danger of explosion (max. temperature 100 °C [212 °F]) and do not attempt to recharge them. Do not open batteries. Only replace by batteries of the same type. Order replacement batteries only from Siemens using the order numbers listed in the Catalog ST 54.1 or in the Appendix in Part 9. You can then be sure that you are using a short-circuit proof battery.

Old batteries with some charge remaining should be discharged with a 10 Ω resistor or torch bulb until no further no-load voltage can be measured.

Completely discharged batteries no longer contain thionyl chloride and are therefore non-toxic and can be disposed of with normal garbage.

Charged lithium thionyl chloride batteries must otherwise be treated as toxic waste.

**Replacing fans**

The service life of the fans (see Section 6.1 "Technical Data") depends on the operating time, ambient temperature and ambient conditions. Resulting damage, e.g. on modules, can be avoided in the event of a fan failure during operation if the fan monitoring is switched on (jumper F-R closed); the power supply unit is then switched off.

In particular circumstances, it may be advisable to replace the fans at corresponding maintenance intervals as a preventive measure.

To replace the fans, proceed as follows:

- Switch off the voltage to the power supply.
- Disassemble the power supply.
- Loosen the fixing screws of the fans.
- Disconnect the plug contacts for the fan power supply.

Insert the fans in the reverse order.

The order nos. of the back-up battery and the various fans can be found in the Appendix.

### 5.4 Pin Assignments of the Power Supply Unit

- The connections of the power supply lines between the power supply unit and the bus PCB are on an 8-pin plug (subminiature plug, 8-pin, fitted with 8 power contacts, series D to MIL-C24308).

For 6ES5 955-3LF12 / 6ES5 955-3NF11 :

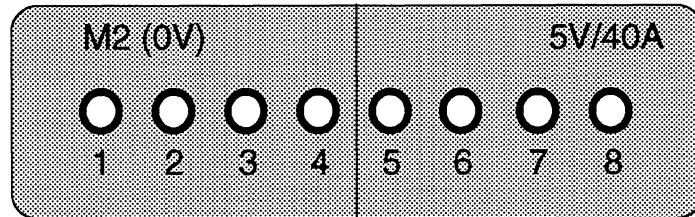


Fig.14 Plug X1, view from rear of device

- The signal connections on the power supply unit are on a 37-pin plug (subminiature plug connector, 37-pin, series D to MIL - C24308).

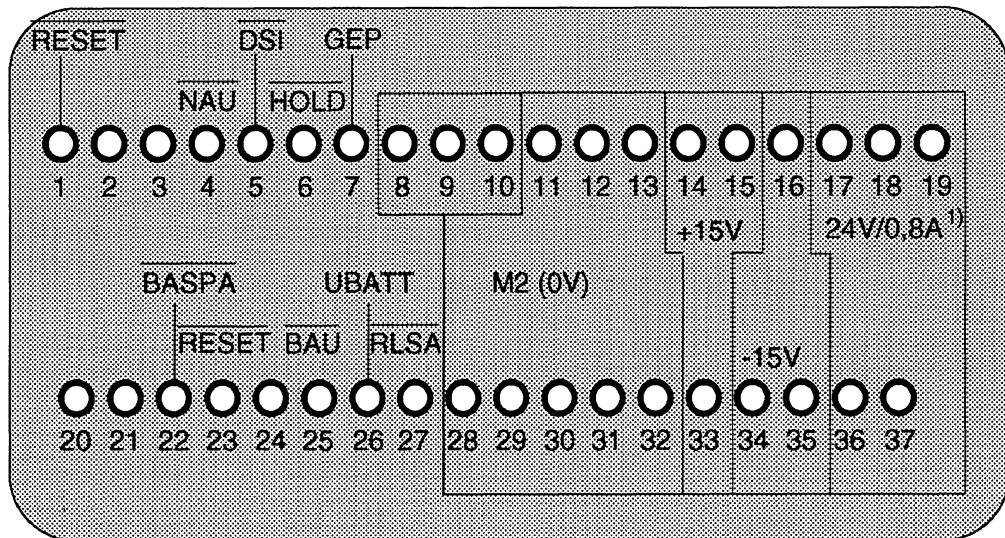


Fig. 15 Plug X2, view from rear of device

1) 2.8 A with power supply unit 6ES5 955-3LF12

5.5 Pin Assignment of the Bus PCB

| Slot 3              |         |        |       |        | Slot 11/51 |         |         |       |        |
|---------------------|---------|--------|-------|--------|------------|---------|---------|-------|--------|
| Backplane Connector | Pin Row |        |       |        | Pin No.    | Pin Row |         |       |        |
|                     | Pin No. | d      | b     | z      |            | f       | d       | b     | z      |
| 1                   | 2       |        | M     | + 5V   | 2          | xLAB0   | M       | M     | + 5V   |
|                     | 4       | UBAT   | PESP  | + 5V   | 4          | xLAB1   | UBAT    | PESP  | + 5V   |
|                     | 6       | GAB12  | GAB0  | CPKL   | 6          | xLAB2   | GAB12   | GAB0  | CPKL   |
|                     | 8       | GAB13  | GAB1  | /GMEMR | 8          | xLAB3   | GAB13   | GAB1  | /GMEMR |
|                     | 10      | GAB14  | GAB2  | /GMEMW | 10         | xLAB4   | GAB14   | GAB2  | /GMEMW |
|                     | 12      | GAB15  | GAB3  | /GRDY  | 12         | xLAB5   | GAB15   | GAB3  | /GRDY  |
|                     | 14      | BUSEN1 | GAB4  | GDB0   | 14         | xLAB6   | /IRA    | GAB4  | GDB0   |
|                     | 16      | BUSEN2 | GAB5  | GDB1   | 16         | xLAB7   | M       | GAB5  | GDB1   |
|                     | 18      | BUSEN3 | GAB6  | GDB2   | 18         | xLAB8   | /xINTMC | GAB6  | GDB2   |
|                     | 20      | BUSEN4 | GAB7  | GDB3   | 20         | xLAB9   | /xINTAS | GAB7  | GDB3   |
|                     | 22      |        | GAB8  | GDB4   | 22         | xLAB10  | /IR E   | GAB8  | GDB4   |
|                     | 24      |        | GAB9  | GDB5   | 24         | xLAB11  | /IR F   | GAB9  | GDB5   |
|                     | 26      |        | GAB10 | GDB6   | 26         | xLAB12  | /IR G   | GAB10 | GDB6   |
|                     | 28      | DSI    | GAB11 | GDB7   | 28         | xLAB13  | DSI     | GAB11 | GDB7   |
|                     | 30      |        | BASP  |        | 30         | xLAB14  | BUSENx  | BASP  |        |
|                     | 32      | /BASPA | M     | /HALT  | 32         | xLAB15  | /BASPA  | M     | /HALT  |

|   |    |       |         |       |    |        |         |         |        |
|---|----|-------|---------|-------|----|--------|---------|---------|--------|
| 2 | 2  |       | M       | + 5V  | 2  | xLDB0  | xLAB16  | M       | + 5V   |
|   | 4  |       | GDB8    | GDB12 | 4  | xLDB1  | xLAB17  | GDB8    | GDB12  |
|   | 6  | /RXD8 | GDB9    | GDB13 | 6  | xLDB2  | M       | GDB9    | GDB13  |
|   | 8  | TXD8  | GDB10   | GDB14 | 8  | xLDB3  | xLAB18  | GDB10   | GDB14  |
|   | 10 | /RXD7 | GDB11   | GDB15 | 10 | xLDB4  | xLAB19  | GDB11   | GDB15  |
|   | 12 | TXD7  | /RXD6   |       | 12 | xLDB5  | /xLMEMR | /xTAU   | M      |
|   | 14 | /RXD5 | TXD6    | /NAU  | 14 | xLDB6  | /xLMEMW | PGBUSX  | /NAU   |
|   | 16 | TXD5  | /RXD4   | /BAU  | 16 | xLDB7  | /xLRDY  | PGBUSY  | /BAU   |
|   | 18 | /RXD3 | TXD4    | + 5V  | 18 | xLDB8  | xNS     | M       | + 5V   |
|   | 20 | TXD3  | /STEU   |       | 20 | xLDB9  | /xHOLDA | /STEU   | /xHOLD |
|   | 22 | /RXD1 | /STOPPA |       | 22 | xLDB10 | TXDx    | /STOPPA | /PEU   |
|   | 24 | TXD1  | /RXD2   | GEP   | 24 | xLDB11 | xHOLDO  | M       | GEP    |
|   | 26 | /TEST | TXD2    |       | 26 | xLDB12 | /TEST   | /RXDx   | /xPARE |
|   | 28 |       | /PERO   | 5V    | 28 | xLDB13 |         | /PERO   | 5V     |
|   | 30 | M24V  | M24V    | M24V  | 30 | xLDB14 | M24V    | M24V    | M24V   |
|   | 32 | 15V   | M       | 24V   | 32 | xLDB15 | 15V     | M       | 24V    |

x = 1 (slot 11)  
x = 2 (slot 51)

Table 8 Pin assignment of the bus PCB (1 of 5)

| Slot 19/59          |         |        |         |         | Slot 27/67 |         |        |         |         |        |
|---------------------|---------|--------|---------|---------|------------|---------|--------|---------|---------|--------|
| Backplane Connector | Pin Row |        |         |         | Pin Row    |         |        |         |         |        |
|                     | Pin No. | f      | d       | b       | z          | Pin No. | f      | d       | b       | z      |
| 1                   | 2       | xLAB0  | M       | M       | + 5V       | 2       | xLAB0  | M       | M       | + 5V   |
|                     | 4       | xLAB1  | UBAT    | PESP    | + 5V       | 4       | xLAB1  | UBAT    | PESP    | + 5V   |
|                     | 6       | xLAB2  | GAB12   | GAB0    | CPKL       | 6       | xLAB2  | GAB12   | GAB0    | CPKL   |
|                     | 8       | xLAB3  | GAB13   | GAB1    | /GMEMR     | 8       | xLAB3  | GAB13   | GAB1    | /GMEMR |
|                     | 10      | xLAB4  | GAB14   | GAB2    | /GMEMW     | 10      | xLAB4  | GAB14   | GAB2    | /GMEMW |
|                     | 12      | xLAB5  | GAB15   | GAB3    | /GRDY      | 12      | xLAB5  | GAB15   | GAB3    | /GRDY  |
|                     | 14      | xLAB6  | /IRA/B  | GAB4    | GDB0       | 14      | xLAB6  | /IRA/B  | GAB4    | GDB0   |
|                     | 16      | xLAB7  | M       | GAB5    | GDB1       | 16      | xLAB7  | M       | GAB5    | GDB1   |
|                     | 18      | xLAB8  |         | GAB6    | GDB2       | 18      | xLAB8  | /xINTMC | GAB6    | GDB2   |
|                     | 20      | xLAB9  |         | GAB7    | GDB3       | 20      | xLAB9  | /xINTAS | GAB7    | GDB3   |
|                     | 22      | xLAB10 | /IR E   | GAB8    | GDB4       | 22      | xLAB10 | /IR E   | GAB8    | GDB4   |
|                     | 24      | xLAB11 | /IR F   | GAB9    | GDB5       | 24      | xLAB11 | /IR F   | GAB9    | GDB5   |
|                     | 26      | xLAB12 | /IR G   | GAB10   | GDB6       | 26      | xLAB12 | /IR G   | GAB10   | GDB6   |
|                     | 28      | xLAB13 | DSI     | GAB11   | GDB7       | 28      | xLAB13 | DSI     | GAB11   | GDB7   |
|                     | 30      | xLAB14 |         | BASP    |            | 30      | xLAB14 | BUSENx  | BASP    |        |
| 32                  | xLAB15  | /BASPA | M       | /HALT   | 32         | xLAB15  | /BASPA | M       | /HALT   |        |
| 2                   | 2       | xLDB0  | xLAB16  | M       | + 5V       | 2       | xLDB0  | xLAB16  | M       | + 5V   |
|                     | 4       | xLDB1  | xLAB17  | GDB8    | GDB12      | 4       | xLDB1  | xLAB17  | GDB8    | GDB12  |
|                     | 6       | xLDB2  | M       | GDB9    | GDB13      | 6       | xLDB2  | M       | GDB9    | GDB13  |
|                     | 8       | xLDB3  | xLAB18  | GDB10   | GDB14      | 8       | xLDB3  | xLAB18  | GDB10   | GDB14  |
|                     | 10      | xLDB4  | xLAB19  | GDB11   | GDB15      | 10      | xLDB4  | xLAB19  | GDB11   | GDB15  |
|                     | 12      | xLDB5  | /xLMEMR | /xTAU   | M          | 12      | xLDB5  | /xLMEMR | /xTAU   | M      |
|                     | 14      | xLDB6  | /xLMEMW | PGBUSX  | /NAU       | 14      | xLDB6  | /xLMEMW | PGBUSX  | /NAU   |
|                     | 16      | xLDB7  | /xLRDY  | PGBUSY  | /BAU       | 16      | xLDB7  | /xLRDY  | PGBUSY  | /BAU   |
|                     | 18      | xLDB8  | xNS     | M       | 5V         | 18      | xLDB8  | xNS     | M       | 5V     |
|                     | 20      | xLDB9  | /xHOLDA | /STEU   | /xHOLD     | 20      | xLDB9  | /xHOLDA | /STEU   | /xHOLD |
|                     | 22      | xLDB10 |         | /STOPPA | /PEU       | 22      | xLDB10 | TXDx    | /STOPPA | /PEU   |
|                     | 24      | xLDB11 | xHOLDO  | M       | GEP        | 24      | xLDB11 | xHOLDO  | M       | GEP    |
|                     | 26      | xLDB12 | /TEST   |         | /xPARE     | 26      | xLDB12 | /TEST   | /RXDx   | /xPARE |
|                     | 28      | xLDB13 |         | /PERO   | 5V         | 28      | xLDB13 |         | /PERO   | 5V     |
|                     | 30      | xLDB14 | M24V    | M24V    | M24V       | 30      | xLDB14 | M24V    | M24V    | M24V   |
| 32                  | xLDB15  | 15V    | M       | 24V     | 32         | xLDB15  | 15V    | 15V     | 24V     |        |

x = 1 (slot 19)  
x = 2 (slot 59)

x = 1 (slot 27)  
x = 2 (slot 67)

Table 8 Pin assignment of the bus PCB (2 of 5)

| Slot 35/43/75/83         |         |        |        |       | Slot 91/99 |         |        |       |        |
|--------------------------|---------|--------|--------|-------|------------|---------|--------|-------|--------|
| Backplane Connector<br>1 | Pin Row |        |        |       | Pin Row    |         |        |       |        |
|                          | Pin No. | f      | d      | b     | z          | Pin No. | d      | b     | z      |
|                          | 2       | xLAB0  | M      | M     | + 5V       | 2       | M      | M     | + 5V   |
|                          | 4       | xLAB1  | UBAT   | PESP  | + 5V       | 4       | UBAT   | PESP  | + 5V   |
|                          | 6       | xLAB2  | GAB12  | GAB0  | CPKL       | 6       | GAB12  | GAB0  | CPKL   |
|                          | 8       | xLAB3  | GAB13  | GAB1  | /GMEMR     | 8       | GAB13  | GAB1  | /GMEMR |
|                          | 10      | xLAB4  | GAB14  | GAB2  | /GMEMW     | 10      | GAB14  | GAB2  | /GMEMW |
|                          | 12      | xLAB5  | GAB15  | GAB3  | /GRDY      | 12      | GAB15  | GAB3  | /GRDY  |
|                          | 14      | xLAB6  | /IR A  | GAB4  | GDB0       | 14      | /IRC/D | GAB4  | GDB0   |
|                          | 16      | xLAB7  | /IR B  | GAB5  | GDB1       | 16      | M      | GAB5  | GDB1   |
|                          | 18      | xLAB8  | /IR C  | GAB6  | GDB2       | 18      |        | GAB6  | GDB2   |
|                          | 20      | xLAB9  | /IR D  | GAB7  | GDB3       | 20      |        | GAB7  | GDB3   |
|                          | 22      | xLAB10 | /IR E  | GAB8  | GDB4       | 22      | /IR E  | GAB8  | GDB4   |
|                          | 24      | xLAB11 | /IR F  | GAB9  | GDB5       | 24      | /IR F  | GAB9  | GDB5   |
|                          | 26      | xLAB12 | /IR G  | GAB10 | GDB6       | 26      | /IR G  | GAB10 | GDB6   |
|                          | 28      | xLAB13 | DSI    | GAB11 | GDB7       | 28      | DSI    | GAB11 | GDB7   |
|                          | 30      | xLAB14 | xSTCK  | BASP  |            | 30      | BUSENx | BASP  |        |
|                          | 32      | xLAB15 | /BASPA | M     | /HALT      | 32      | /BASPA | M     | /HALT  |

|                          |        |        |         |         |        |     |       |         |       |
|--------------------------|--------|--------|---------|---------|--------|-----|-------|---------|-------|
| Backplane Connector<br>2 | 2      | xLDB0  | xLAB16  | M       | + 5V   | 2   |       | M       | + 5V  |
|                          | 4      | xLDB1  | xLAB17  | GDB8    | GDB12  | 4   |       | GDB8    | GDB12 |
|                          | 6      | xLDB2  | M       | GDB9    | GDB13  | 6   | M     | GDB9    | GDB13 |
|                          | 8      | xLDB3  | xLAB18  | GDB10   | GDB14  | 8   |       | GDB10   | GDB14 |
|                          | 10     | xLDB4  | xLAB19  | GDB11   | GDB15  | 10  |       | GDB11   | GDB15 |
|                          | 12     | xLDB5  | /xLMEMR |         | M      | 12  |       |         | M     |
|                          | 14     | xLDB6  | /xLMEMW | PGBUSX  | /NAU   | 14  |       | PGBUSX  | /NAU  |
|                          | 16     | xLDB7  | /xLRDY  | PGBUSY  | /BAU   | 16  |       | PGBUSY  | /BAU  |
|                          | 18     | xLDB8  |         | M       | 5V     | 18  |       | M       | 5V    |
|                          | 20     | xLDB9  |         | /STEU   |        | 20  |       | /STEU   |       |
|                          | 22     | xLDB10 |         | /STOPPA |        | 22  | TXDx  | /STOPPA | /PEU  |
|                          | 24     | xLLB11 |         | M       | GEP    | 24  |       | M       | GEP   |
|                          | 26     | xLDB12 | /TEST   |         | /xPARE | 26  | /TEST | /RXDx   |       |
|                          | 28     | xLDB13 |         | /PERO   | 5V     | 28  |       | /PERO   | 5V    |
|                          | 30     | xLDB14 | M24V    | M24V    | M24V   | 30  | M24V  | M24V    | M24V  |
| 32                       | xLDB15 | 15V    | M       | 24V     | 32     | 15V | M     | 24V     |       |

x = 1 (slots 35 and 43)  
x = 2 (slots 75 and 83)

x = 1 (slot 91)  
x = 2 (slot 99)

Table 8 Pin assignment of the bus PCB (3 of 5)

| Slot 107/115/123/131 |         |        |       |        | Slot 139/147 |         |       |        |  |
|----------------------|---------|--------|-------|--------|--------------|---------|-------|--------|--|
| Backplane Connector  | Pin Row |        |       |        | Pin No.      | Pin Row |       |        |  |
|                      | Pin No. | d      | b     | z      |              | d       | b     | z      |  |
| 1                    | 2       | M      | M     | +5V    | 2            |         | M     | 5V     |  |
|                      | 4       | UBAT   | PESP  | +5V    | 4            | (UBAT)  | PESP  | 5V     |  |
|                      | 6       | GAB12  | GAB0  | CPKL   | 6            | GAB12   | GAB0  | CPKL   |  |
|                      | 8       | GAB13  | GAB1  | /GMEMR | 8            | GAB13   | GAB1  | /GMEMR |  |
|                      | 10      | GAB14  | GAB2  | /GMEMW | 10           | GAB14   | GAB2  | /GMEMW |  |
|                      | 12      | GAB15  | GAB3  | /GRDY  | 12           | GAB15   | GAB3  | /GRDY  |  |
|                      | 14      | /IRA   | GAB4  | GDB0   | 14           | (/IRA)  | GAB4  | GDB0   |  |
|                      | 16      | /IRB   | GAB5  | GDB1   | 16           | (/IRB)  | GAB5  | GDB1   |  |
|                      | 18      | /IR C  | GAB6  | GDB2   | 18           | (/IRC)  | GAB6  | GDB2   |  |
|                      | 20      | /IR D  | GAB7  | GDB3   | 20           | (/IRD)  | GAB7  | GDB3   |  |
|                      | 22      | /IR E  | GAB8  | GDB4   | 22           | (/IRE)  | GAB8  | GDB4   |  |
|                      | 24      | /IR F  | GAB9  | GDB5   | 24           | (/IRF)  | GAB9  | GDB5   |  |
|                      | 26      | /IR G  | GAB10 | GDB6   | 26           | (/IRG)  | GAB10 | GDB6   |  |
|                      | 28      | DSI    | GAB11 | GDB7   | 28           | (DSI)   | GAB11 | GDB7   |  |
|                      | 30      |        | BASP  |        | 30           | M       | BASP  |        |  |
|                      | 32      | /BASPA | M     | /HALT  | 32           | /BASPA  | M     |        |  |

|   |    |      |         |       |    |        |           |        |
|---|----|------|---------|-------|----|--------|-----------|--------|
| 2 | 2  |      | M       | +5V   | 2  |        | M         | +5V    |
|   | 4  |      | GDB8    | GDB12 | 4  |        | GDB8      | GDB12  |
|   | 6  | M    | GDB9    | GDB13 | 6  |        | GDB9      | GDB13  |
|   | 8  |      | GDB10   | GDB14 | 8  |        | GDB10     | GDB14  |
|   | 10 |      | GDB11   | GDB15 | 10 |        | GDB11     | GDB15  |
|   | 12 |      |         | M     | 12 |        |           |        |
|   | 14 |      | PGBUSX  | /NAU  | 14 |        |           | (/NAU) |
|   | 16 |      | PGBUSY  | /BAU  | 16 |        |           | (/BAU) |
|   | 18 |      | M       | 5V    | 18 |        | /PEU      | /CPKLA |
|   | 20 |      |         |       | 20 |        |           |        |
|   | 22 | TXDx | /STOPPA |       | 22 |        | (/STOPPA) | M      |
|   | 24 |      | M       | GEP   | 24 |        | M         | (GEP)  |
|   | 26 |      | /RXDx   |       | 26 |        |           | M      |
|   | 28 |      | /PERO   | 5V    | 28 |        | M         | M      |
|   | 30 | M24V | M24V    | M24V  | 30 | (M24V) | (M24V)    | (M24V) |
|   | 32 | 15V  | M       | 24V   | 32 | (15V)  | M         | (24V)  |

x = 5 (slot 107)  
 x = 6 (slot 115)  
 x = 7 (slot 123)  
 x = 8 (slot 131)

( ) only if jumpers 1 - 16 are closed  
 (not as delivered)

Table 8 Pin assignment of the bus PCB (4 of 5)

| Slot 155/163        |         |        |       |        |
|---------------------|---------|--------|-------|--------|
| Backplane Connector | Pin Row |        |       |        |
|                     | Pin No. | d      | b     | z      |
| 1                   | 2       |        | M     | 5V     |
|                     | 4       | 5V     | PESP  | 5V     |
|                     | 6       | GAB12  | GAB0  | CPKL   |
|                     | 8       | GAB13  | GAB1  | /GMEMR |
|                     | 10      | GAB14  | GAB2  | /GMEMW |
|                     | 12      | GAB15  | GAB3  | /GRDY  |
|                     | 14      | 5V     | GAB4  | GDB0   |
|                     | 16      | 5V     | GAB5  | GDB1   |
|                     | 18      |        | GAB6  | GDB2   |
|                     | 20      |        | GAB7  | GDB3   |
|                     | 22      | M      | GAB8  | GDB4   |
|                     | 24      | M      | GAB9  | GDB5   |
|                     | 26      | M      | GAB10 | GDB6   |
|                     | 28      |        | GAB11 | GDB7   |
|                     | 30      | M      | BASP  |        |
|                     | 32      | /BASPA | M     |        |

|                          |    |  |       |        |
|--------------------------|----|--|-------|--------|
| Backplane Connector<br>2 | 2  |  | M     | 5V     |
|                          | 4  |  | GDB8  | GDB12  |
|                          | 6  |  | GDB9  | GDB13  |
|                          | 8  |  | GDB10 | GDB14  |
|                          | 10 |  | GDB11 | GDB15  |
|                          | 12 |  | 5V    | 5V     |
|                          | 14 |  | 5V    | 5V     |
|                          | 16 |  | 5V    | 5V)    |
|                          | 18 |  | /PEU  | /CPKLA |
|                          | 20 |  |       |        |
|                          | 22 |  |       | M      |
|                          | 24 |  | M     |        |
|                          | 26 |  |       | M      |
|                          | 28 |  | M     | M      |
|                          | 30 |  |       |        |
|                          | 32 |  | M     |        |

Table 8 Pin assignment of the bus PCB (5 of 5)



## 5.6 Pin Designation of the Interrupt Signals on the Bus PCB (only for S5-155U Programmable Controller)

| Module   | Interrupt sink* |       |       |       | Interrupt source* |       |   |               |
|----------|-----------------|-------|-------|-------|-------------------|-------|---|---------------|
|          | CPU 1           | CPU 2 | CPU 3 | CPU 4 | I/Os / CP         |       |   |               |
| Slot no. | 11              | 51    | 91    | 99    |                   |       | 35, 43,<br>75, 83,<br>107, 115,<br>123, 131 | 139,147<br>** |
| Signal   |                 |       |       |       |                   |       |   |               |
| IRA      | 1d 14           |       |       |       | 1d 14             |       | 1d 14                                       | 1d 14         |
| IRB      |                 | 1d 14 |       |       |                   | 1d 14 | 1d 16                                       | 1d 16         |
| IRC      |                 |       | 1d 14 |       |                   |       | 1d 18                                       | 1d 18         |
| IRD      |                 |       |       | 1d 14 |                   |       | 1d 20                                       | 1d 20         |
| IRE      | 1d 22           | 1d 22 | 1d 22 | 1d 22 | 1d 22             | 1d 22 | 1d 22                                       | 1d 22         |
| IRF      | 1d 24           | 1d 24 | 1d 24 | 1d 24 | 1d 24             | 1d 24 | 1d 24                                       | 1d 24         |
| IRG      | 1d 26           | 1d 26 | 1d 26 | 1d 26 | 1d 26             | 1d 26 | 1d 26                                       | 1d 26         |

Table 9 Pin designation of the interrupt signals (on connector X1)

\* Designations according to ISO 2382/XVI - 1978 (DIN 44301)

Interrupt sink = module that receives the interrupt

Interrupt source = module that generates the interrupt

\*\* These slots can only be used for PE modules with interrupt outputs, when the jumpers 7-13 are inserted in the bus PCB. (The jumpers are not inserted when delivered.)

Please set the necessary jumpers on the modules if you use the interrupt signals. The settings are described in the Instructions for CPU and I/O modules, whereas for the CPUs, only the CPU 946/947 has jumpers.

Also refer to the CPU 946/947 Programming Guide for notes on programming (Section "Interrupt Driven Program Processing").

### Note



Interrupt processing is not possible in the S5-155H programmable controller (Interrupt processing only via EB 0).

## 6 Technical Data of the S5-155U Central Controller

This power supply is UL and CSA listed.

| <b>Device safety</b>   |  |
|--|--|
| Device complies with   | VDE 0160. Protection against overvoltage complying with VDE 0160 A1 (April 89), Section 6.3.4 (overvoltage proof) achievable with additional measures.   |
| Protection class   | I<br>(safe electrical isolation of the primary and secondary circuit of the power supply unit)   |
| Degree of protection   | IP 20 to IEC 529/DIN 40050 with covering of empty slots by dummy front panels  |
| <b>Climatic ambient conditions (tested according to IEC 68-2/-1/-2/-3)</b>   |  |
| Temperature:<br>Operation<br>- Device freely installed, supply air temperature at lower air inlet of the power supply<br><br>- Device installed in cabinet<br><br>Transport and storage temperature<br><br>Temperature change<br>- operation<br>- transport and storage<br><br><b>Relative humidity</b><br>- operation<br>- transport and storage<br><br><b>Operating altitude</b><br>- operation<br>- transport and storage<br><br><b>Toxic substances</b><br>SO <sub>2</sub><br>H <sub>2</sub> S | 0 to 55 °C [32 to 131 °F]<br><br>(When installing in a cabinet, take into account that the dissipated heat loss depends on the cabinet design, its ambient temperature and the device arrangement))<br><br>-40 to +70 °C [-40 to +158 °F]<br><br>max. 10 K/h<br>max. 20 K/h<br>(3 h adaption time when delivered below 0 °C [+32 °F] because of possible condensation)<br><br>max. 95 % at 25 °C [77 °F], no condensation<br>max. 95 % at 25 °C [77 °F], no condensation<br><br>- 1000 m to + 1500 m <sup>1)</sup><br>- 1000 m to + 3500 m<br><br>max. 0.5 ppm (relative humidity below 60 %)<br>max. 0.1 ppm (relative humidity below 60 %) |

1) Note, if the programmable controller is used above 1500 m:  
It is advisable to contact your Siemens representative regarding the required cooling conditions.

Table 10 Technical data (1 of 6)

| <b>Mechanical ambient conditions (tested according to IEC 68-2-6)</b> |   |
|---|---|
| Vibrations during operation   | 10 to 58 Hz (constant amplitude 0.15 mm)<br>58 to 500 Hz (constant acceleration 2 g)  |
| <b>Noise immunity (EMC)</b>   |   |
| Radio interference suppression  | to DIN VDE 0871 (CISPR, Publication No. 11 and CENELEC, HD 344)   |
| Limit class   | A   |
| Conducted interferences on:<br>AC voltage supply lines (230 V AC)     | 2 kV to IEC 801-4 (burst)<br><br>1 kV to IEC 801-5<br>line against line ( $\mu$ s pulses)<br><br>2 kV to IEC 801-5<br>line against ground ( $\mu$ s pulses)             |
| DC voltage supply lines<br>(24 V-supply)                              | 1 kV to IEC 801-4 (burst)   |
| Signal lines (24 V DC)  | 1 kV to IEC 801-4 (burst)   |
| Signal lines (230 V AC)   | 2 kV to IEC 801-4 (burst)   |
| Noise immunity to discharges of static electricity                    | A noise immunity of 8 kV must be guaranteed by an appropriate design  |
| Noise immunity with respect to RF radiation                           | RF radiation according to VG 95373 LFO2G limit class 3 (up to 200 MHz), corresponding to 3 V/m  |
| <b>Back-up battery</b>  |   |
| Type  | Lithium-thionyl-chloride  |
| Capacity  | 5 Ah at $I_{max} = 35$ mA   |
| No-load voltage   | 3.6 V   |
| Operating voltage   | 3.4 V   |
| Storage life  | approx. 10 years  |
| Service life during operation   | max. 3 years  |
| <b>Mechanical design</b>  |   |
| Mechanical requirements   | Installation in fixed devices not free from vibrations; installation on ships and vehicles with observance of special installation specifications, but not on the motor |
| Weight  | approx. 14 kg   |
| Dimensions (w x h x d)  | 482.6 x 432 x 310 mm  |

Table 10 Technical data (2 of 6)

## 6.1 Power Supply Unit 6ES5 955-3LF12

This power supply unit is UL and CSA listed.

| <b>Input</b>   |  |
|--|--|
| Rated input voltage $U_{EN}$   | 230/120 V AC + 10 %/- 18.7 % <sup>1)</sup>   |
| Undervoltage signal $U_E$  | < 187 V AC (or 93 V AC)  |
| Input frequency $f_E$  | 48 to 63 Hz  |
| Input current $I_{EN}$<br>with rated load and<br>$U_{EN} = 240$ V (or 120 V) | 2.4 A (4.8 A)  |
| Inrush current peak $I_{Emax}$   | 200 A (100 A)  |
| Efficiency with rated load, with fan   | typically $\geq 70$ %  |
| Stored energy time during power failure                                      | > 5 ms   |
| Power factor $\cos \varphi$  | 0.73   |
| Input fuse (internal)  | 6 A fast-blow; 250 V; 6.3 x 32 mm ;<br>location F107 (printed on power supply<br>board)      |
| Electrical isolation   | yes  |
| <b>Output 1</b>  |  |
| Rated output voltage $U_{AN1}$   | 5.1 V DC $\pm 0.5$ %   |
| Setting range of output voltage  | (0.95 to 1.05) x $U_{AN1}$   |
| Rated output current $I_{AN1}$   | 40 A DC  |
| Ripple   | $\leq 1$ % of $U_{A1}$   |
| Dynamic voltage tolerance<br>with load surge from 50 to 100 % $I_N$          | $\leq 5$ % of $U_{A1}$   |
| Settling time  | $\leq 5$ ms  |
| Overvoltage shut-down $U_{A1}$   | 6 V $\pm 5$ %  |
| Undervoltage signal $U_{A1}$   | 4.75 V + 5 %   |
| Current limiting with overload   | (1.05 to 1.15) x $I_{AN1}$   |
| <b>Output 2</b>  |  |
| Rated output voltage $U_{AN2}$   | 24 V DC + 25 %/- 17 %  |
| Rated output current $I_{AN2}$   | 2.8 A DC <sup>2)</sup>   |
| Total current load of the 24 V and<br>15 V output                            | $\leq 2.8$ A   |
| Ripple   | $\leq 5$ % of $U_{A2}$   |
| Fuse for overcurrent protection  | 4 A fast-blow ; 250 V ; 6.3 mm x 32 mm ;<br>location F255 (printed on power supply<br>board) |

1) Voltage selector

2) Total output currents ( $I_{A2} + I_{A3} + I_{A4}$ )  $\leq 2.8$  A DC

Table 10 Technical data (3 of 6)

| <b>Output 3 with supplementary module</b>  |   |
|--|---|
| Rated output voltage $U_{AN3}$<br>Rated output current $I_{AN3}$<br>Ripple<br>Overvoltage shut-down<br>Undervoltage signal<br>(LED on front panel)<br>Overcurrent protection $I_{A3}$<br>by current limiting | 15 V DC $\pm$ 5 %<br>2 A DC <sup>1)</sup><br>$\leq$ 5 % of $U_{AN3}$<br>$U_{A3} \geq 18.5$ V<br>$U_{A3} \leq 14$ V $\pm$ 3 %<br>2 to 3 A  |
| <b>Output 4</b>  |   |
| Rated output voltage $U_{AN4}$<br>Rated output current $I_{AN4}$<br>with output 2 and 3 omitted<br>Current limiting (reaction threshold)<br>Undervoltage signal<br>(LED on front panel)<br>Capacitive load   | 24 V DC + 6 V/- 5 V<br><br>0.4 A <sup>1)</sup><br>$\geq 0.44$ A<br>16 V $\pm$ 20 %<br><br>max. 100 nF   |
| <b>Fans</b>  |   |
| Fan type<br>Input voltage<br>Delivery rate per fan<br>Service life of fan<br><br><br>Fan monitoring  | 2 axial fans<br>240/120 V AC, selectable<br>160 m <sup>3</sup> /h (no load)<br>typically 30 000 to 40 000 h at 55 °C<br>[131 °F]<br>typically 40 000 to 50 000 h at 30 °C<br>[86 °F]<br>Flow monitoring with thermistors as sensors; stoppage at 1 or both fans is recognized and signalled externally by LEDs and relay contacts and the output voltages are switched off. (Jumper F-R on power supply closed.) If jumper F-R is open, only the relay contact responds in the event of a fault and the power supply must be switched off externally. |
| <b>Additional monitoring</b>   |   |
| 24 V load voltage (external voltage monitor)   | $\geq 14$ to 20 V   |
| Electrical isolation primary/secondary   | yes   |

1) Total of output currents ( $I_{AN2} + I_{AN3} + I_{AN4}$ )  $\leq$  2.8 DC

Table 10 Technical data (4 of 6)

## 6.2 Power Supply Unit 6ES5 955-3NF11

This power supply unit is UL and CSA listed.

| <b>Input</b>  |   |
|---|---|
| Rated input voltage $U_{EN}$  | 24 V DC + 25 %/-17 %  |
| Undervoltage signal $U_E$   | < 20 V DC   |
| Input frequency $f_E$   | -   |
| Input current $I_{EN}$<br>with rated load and<br>$U_{EN} = 24$ V DC | 17.5 A  |
| Inrush current peak $I_{Emax}$                                      | 300 A   |
| Efficiency with rated load, with fan                                | typically 65 %  |
| Stored energy time during power failure                             | > 5 ms  |
| Input fuse (internal)   | 30 A/medium time-lag; 250 V; 6.3x32 mm;<br>location F110 (printed on power supply<br>board) |
| Electrical isolation  | yes   |
| <b>Output 1</b>   |   |
| Rated output voltage $U_{AN1}$                                      | 5.1 V DC $\leq 0.5$ %   |
| Setting range of output voltage                                     | (0.95 to 1.05) x $U_{AN1}$  |
| Rated output current $I_{AN1}$                                      | 40 A DC   |
| Ripple  | $\leq 1$ % of $U_{A1}$  |
| Dynamic voltage tolerance<br>with load surge from 50 to 100 % $I_N$ | $\leq 5$ % of $U_{A1}$  |
| Settling time   | $\leq 5$ ms   |
| Overvoltage shut-down $U_{A1}$                                      | 6 V $\pm 5$ %   |
| Undervoltage signal $U_{A1}$  | 4.75 V $\pm 5$ %  |
| Current limiting with overload                                      | (1.05 to 1.15) x $I_{AN1}$  |
| <b>Output 2</b>   |   |
| Rated output voltage $U_{AN2}$                                      | 24 V DC + 25 %/- 17 %   |
| Rated output current $I_{AN2}$ <sup>1)</sup>                        | 2.8 A DC  |
| Total current load of the 24 V and 15 V output                      | $\leq 28$ A   |
| Ripple  | $\leq 5$ % of $U_{A2}$  |
| Fuse for overcurrent protection                                     | 4 A fast-blow; 250 V ; 6.3 x 32 mm<br>location F177 (printed on power supply<br>board)      |
| <b>Output 3 with supplementary module</b>                           |   |
| Rated output voltage $U_{AN3}$                                      | 15 V DC $\pm 5$ %   |
| Rated output current $I_{AN3}$ <sup>1)</sup>                        | 2 A DC  |
| Ripple  | $\leq 5$ % of $U_{AN3}$   |
| Overvoltage shut-down   | $U_{A3} \geq 18.5$ V  |
| Undervoltage signal<br>(LED on front panel)                         | $U_{A3} \leq 14$ V $\pm 3$ %  |
| Overcurrent protection $I_{A3}$<br>by current limiting              | 2 to 3 A  |

1) Total of output currents ( $I_{AN2} + I_{AN3} + I_{AN4}$ )  $\leq 2.8$  A DC

Table 10 Technical data (5 of 6)

| <b>Output 4 : 24 V-Front</b>  |   |
|---|---|
| Rated output voltage $U_{AN4}$<br>Rated output current $I_{AN4}$ <sup>1)</sup><br>Current limiting (reaction threshold)<br>Undervoltage signal<br>(LED on front panel)<br>Capacitive load | 24 V DC (+ 6 V/- 5 V)<br>0.4 A<br>$\geq 0.44$ A<br>16 V $\pm$ 20 %<br>max. 100 nF   |
| <b>Fans</b>   |   |
| Fan type<br>Input voltage<br>Delivery rate per fan<br>Service life of fan<br><br>Fan monitoring   | 2 axial fans<br>24 V DC<br>160 m <sup>3</sup> /h (no load)<br>typically 30 000 to 40 000 h at 55 °C [131 °F]<br>typically 40 000 to 50 000 h at 30 °C [86 °F]<br>Flow monitoring with thermistors as sensors;<br>stoppage of 1 or both fans is recognized and<br>signalled externally by LEDs and relay<br>contacts and the output voltages are<br>switched off (jumper F-R on power supply<br>closed). If jumper F-R is open, only the relay<br>contact responds in the event of a fault and<br>the power supply must be switched off<br>externally. |
| <b>Additional monitoring</b>  |   |
| 24 V load voltage (external voltage monitor)  | $\geq 14$ to 20 V   |
| Electrical isolation primary/secondary  | yes   |

1) Total of output currents ( $I_{A2} + I_{A3} + I_{A4}$ )  $\leq 2.8$  A DC

Table 10 Technical data (6 of 6)

## Index

## A

Ambient conditions ..... 6-1

## B

Back-up battery ..... 5-3, 6-2

Battery failure ..... 3-4

Battery voltage ..... 4-3

Bus PCB ..... 1-4

## C

Cable duct ..... 1-4

Coordinator ..... 1-13 - 1-14

CSA ..... 6-1, 6-3, 6-5

## D

Degree of protection ..... 6-1

Device safety ..... 6-1

## E

EN input ..... 3-1

Enable circuit ..... 5-1

Enable Power Supply ..... 3-1

Enable voltage ..... 5-1

Equipotential bonding conductor ..... 1-13

## F

Fans ..... 6-4, 6-6

Fibre-optic interface module ..... 1-13

Front plug ..... 2-4

Fuse ..... 4-5

## H

Heat exchanger ..... 3-2

## I

I/O address ..... 1-7

I/O area ..... 1-7

Individual locking mechanism ..... 1-4

Input fuse ..... 4-5

Installation specifications ..... 6-2

Interface cable ..... 1-9

Interrupt signals ..... 5-11

## L

Locking bar ..... 1-4



**M**

Maintenance intervals..... 5-4  
Mechanical design ..... 6-2  
Monitor Output ..... 3-1, 3-4  
Multiplexer function..... 1-5, 1-14  
Multiprocessor operation ..... 1-13, 4-9

**N**

Noise immunity ..... 6-2

**O**

O (extended) I/O area..... 1-7  
O page number..... 1-7  
Output fuse ..... 4-5

**P**

Power supply unit ..... 1-4, 6-3, 6-5  
Protection class ..... 6-1

**R**

Reset ..... 4-3

**S**

Screen connection ..... 1-11  
Signalling relay ..... 3-4  
Single processor operation ..... 1-13, 4-8  
Supplementary module..... 2-3

**T**

Temperature monitoring device ..... 3-2  
Terminator ..... 1-9  
Thermostat..... 3-2

**U**

UL ..... 6-1, 6-3, 6-5

**V**

Voltage Monitor ..... 3-1, 4-2

**W**

Winchester disk drive ..... 3-3

# SIEMENS

## SIMATIC S5

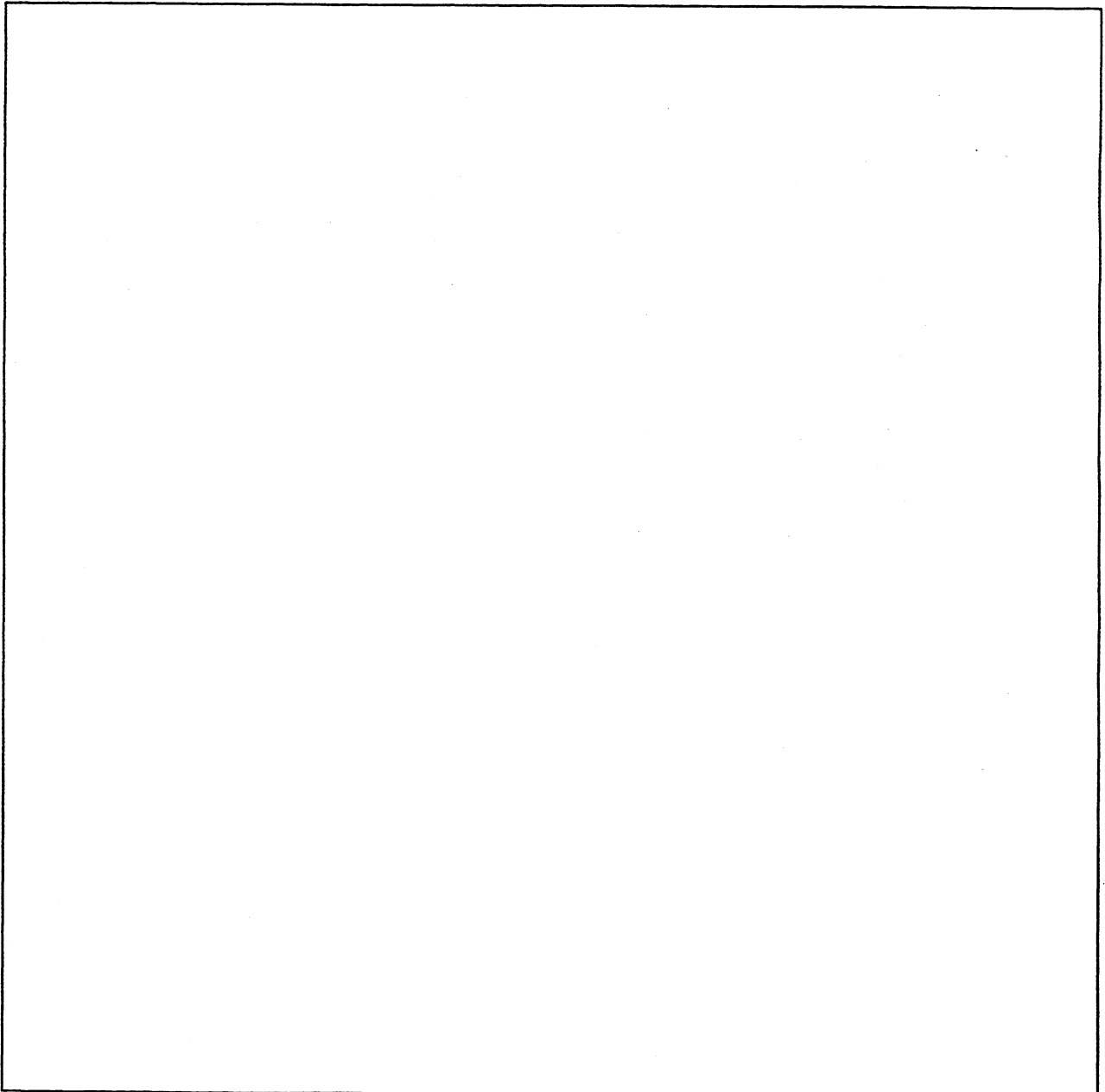
CPU 946/947

6ES5 946-3UA21/22/23

6ES5 947-3UA21/22/23

Instructions

C79000-B8576-C532-06





---

Siemens has developed this document for its licensees and customers. The information contained herein is the property of Siemens and may not be copied, used, or disclosed to others without prior written approval from Siemens. Users are cautioned that the material contained herein is subject to change by Siemens at any time and without prior notice.

Siemens shall not be responsible for any damages, including consequential damages, caused by reliance on material presented, including but not limited to typographical, electronic, arithmetic, or listing errors.

---

|   |  |
|---|--|
|  |  <b>WARNING</b>   |
|   | <p><b>Hazardous voltage.</b></p> <p><b>Can cause death, severe personal injury, or substantial property damage.</b></p> <p>Restrict use to qualified personnel.<br/>See safety instructions.</p> |

Only qualified personnel should install or maintain this equipment after becoming thoroughly familiar with all warnings, safety notices, and maintenance procedures contained in this manual. The successful and safe operation of this equipment is dependent upon proper handling, installation, operation, and maintenance.

The following are definitions of the terms "qualified person," "danger," "warning," and "caution," as applicable for this document.

**Qualified Person**

One who is familiar with the installation, construction, and operation of this equipment and the hazards involved. In addition, the person should have the following qualifications:

- Be trained and authorized to use and tag circuits and equipment in accordance with established safety practices
- Be trained in the proper care and use of protective equipment in accordance with established safety practices
- Be trained in rendering first aid

**DANGER**

Indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.

**WARNING**

Indicates that loss of life, 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.

---

STEP 5® and SIMATIC® are registered trademarks of Siemens AG.

---

Copyright © Siemens AG 1991  
First Printing, September 1989  
Printed in the Federal Republic of Germany

---

## **Preface**

CPU 946/947 is the standard central processing unit for the SIMATIC S5-155U programmable controller.

This book describes the hardware and installation procedure for CPU 946/947 and lists its technical specifications.

This book is intended for engineers, programmers, and maintenance personnel who have a general knowledge of programmable controller concepts.

If you have any questions about CPU 946/947 not answered in this book, please contact your local Siemens representative.



## How to Use This Book

This section discusses information that may be helpful as you use this book.

### Contents of This Book

- **Chapter 1 - Technical Description**

This chapter describes the application and design of CPU 946/947 and includes its memory assignment and contains a list of technical data.

- **Chapter 2 - Installation and Operator Information**

This chapter explains how to install and remove CPU 946/947. It also explains the control switches and LEDs on the front panel of CPU 946/947.

- **Chapter 3 - Operation**

This chapter explains the restart procedure of CPU 946/947 and defines its restart types. It also discusses the programmer interface and the operation of peripheral modules.

- **Chapter 4 - Maintenance**

This chapter explains the central register for error addresses of CPU 946/947. It also shows the layout of the CPU jumpers and describes the interface assignment for its backplane connectors and front connectors.

- **Index**

The index contains an alphabetical list of key terms and subjects covered in this book and their corresponding page numbers.

- **Remarks Form**

The remarks form is provided for your comments and recommendations.

- **Training**

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

- **Reference Materials**

It is recommended that you have the following books that support the S5-155U system:

- *Catalog ST 54.1: S5-135U, S5-155U and S5-155H Programmable Controllers*  
(Order No. E86010-K4654-A111-A6-7600)<sup>1</sup>
- Programmer Manuals:
  - *PG 685 Programmer Manual*  
(Order No. 6ES5 885-0SC21)\*
  - *PG 710 Programmer Manual*  
(Order No. 6ES5 814-0MC21)\*
  - *PG 730 Programmer Manual*  
(Order No. 6ES5 834-0FC21)\*
  - *PG 750 Programmer Manual*  
(Order No. 6ES5 886-0FC21)\*
  - *PG 750-486 Programmer Manual*  
(Order No. 6ES5 886-0FC22)\*
  - *PG 770 Programmer manual*  
(Order No. 6ES5 887-0FC21)\*
  - *STEP 5 for Personal Computers User Guide*  
(Order No. 6ES5 896-0SC21)\*
  - *S5-135U (CPU 928B) Manual*  
(Order No. 6ES5 998-2UL22)\*
  - *S5-135U (CPU 928) Manual*  
(Order No. 6ES5 998-1UL23)\*
  - *S5-135U (CPU 921/922) Manual*  
(Order No. 6ES5 998-0UL22)\*
  - *U Periphery Manual*  
(Order No. 6ES5 998-0PC22)\*

You will find an introduction to programming with STEP 5, as well as an explanation of how to work with the S5-155U programmable controller and its I/O modules in the following book...

*Automating with the SIMATIC S5-155U*  
by Hans Berger  
Siemens AG, ISBN 3-8009-1562-6

---

<sup>1</sup> Order the appropriate book from your local Siemens representative.

**Conventions**

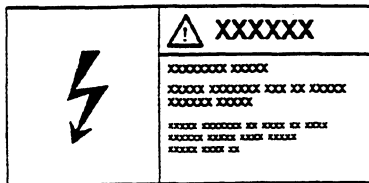
The following conventions are used in this book and are listed for your reference:

**Convention**

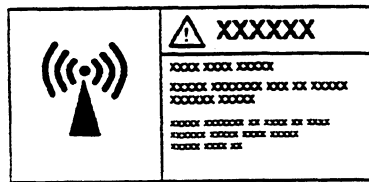
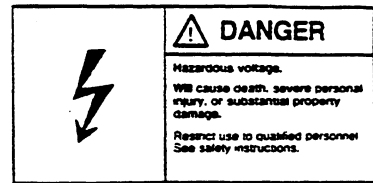
**Definition**

**Example**

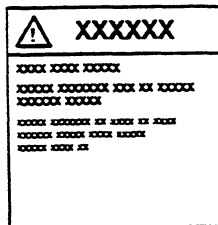
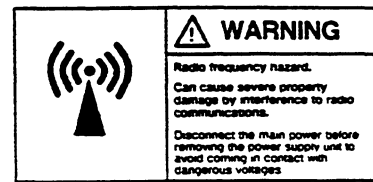
A box that indicates a type of hazard, describes its implications, and tells you how to avoid the hazard is safety notation. Some safety notation includes a graphic symbol representing an electrical or radio frequency hazard. All safety notation has one of the following levels of caution:



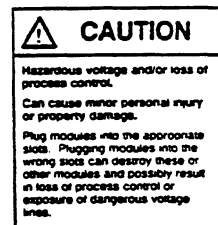
- A danger indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.



- A warning indicates that loss of life, severe personal injury, or substantial property damage can result if proper precautions are not taken.



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





# Contents

|  |         |
|--|---------|
| <b>Preface</b> .....                                       | 0-1     |
| <b>How to Use This Book</b> .....                          | 0-3     |
| <b>Chapter 1 - Technical Description</b>                   |         |
| 1.1 Application .....                                      | 1-1     |
| 1.2 Design .....   | 1-1     |
| 1.3 System Structure .....                                 | 1-2     |
| 1.4 Memory Assignment .....                                | 1-5     |
| 1.5 Technical Specifications .....                         | 1-6     |
| 1.6 Functional Capabilities .....                          | 1-8     |
| <b>Chapter 2 - Installation and Operator Information</b>   |         |
| 2.1 Plugging In and Removing Modules .....                 | 2-1     |
| 2.2 Control Switches and LEDs .....                        | 2-2     |
| 2.2.1 "RUN/STOP" Switch .....                              | 2-4     |
| 2.2.2 "RESET" Switch .....                                 | 2-4     |
| 2.2.3 Operating Status LEDs .....                          | 2-5     |
| 2.2.4 Error and Signal LEDs .....                          | 2-7     |
| 2.3 Restart Procedure for the CPU 946/947 .....            | 2-9     |
| 2.4 Operation of Peripheral Modules .....                  | 2-10    |
| <b>Chapter 3 - Operation</b>                               |         |
| 3.1 Restart Types .....                                    | 3-1     |
| 3.1.1 Cold Restart .....                                   | 3-1     |
| 3.1.2 Manual Warm Restart .....                            | 3-1     |
| 3.1.3 Automatic Warm Restart .....                         | 3-2     |
| 3.1.4 Automatic Cold Restart .....                         | 3-2     |
| 3.1.5 Test Mode .....                                      | 3-2     |
| 3.2 Programmer Interface .....                             | 3-2     |
| <b>Chapter 4 - Maintenance</b>                             |         |
| 4.1 Central Register for Error Addresses .....             | 4-1     |
| 4.2 Jumper Layout .....                                    | 4-2     |
| 4.3 Interface Assignment of the Backplane Connectors ..... | 4-8     |
| 4.4 Interface Assignment of the Front Connector .....      | 4-9     |
| <b>Index</b> .....   | Index-1 |

**Figures**

|     |  |     |
|-----|--|-----|
| 1-1 | Block Diagram of CPU 946/947 with Optional 355 Memory Module | 1-2 |
| 2-1 | Front Panel of CPU 946/947                                   | 2-3 |
| 4-1 | Jumper Layout on CPU 946 (6ES5 946-3UA21)                    | 4-2 |
| 4-2 | Jumper Layout on CPU 947 (6ES5 947-3UA21)                    | 4-3 |
| 4-3 | Jumper Layout on CPU 946 (6ES5 946-3UA22)                    | 4-4 |
| 4-4 | Jumper Layout on CPU 947 (6ES5 947-3UA22)                    | 4-5 |
| 4-5 | Jumper Layout on CPU 946 (6ES5 946-3UA23)                    | 4-6 |
| 4-6 | Jumper Layout on CPU 947 (6ES5 947-3UA23)                    | 4-7 |

**Tables**

|     |   |     |
|-----|---|-----|
| 1-1 | CPU 946/947 Memory Assignment                     | 1-5 |
| 2-1 | CPU 946/947 "RESET" Switch Settings               | 2-3 |
| 4-1 | Interface Assignment of the Backplane Connector 1 | 4-8 |
| 4-2 | Interface Assignment of the Backplane Connector 2 | 4-8 |
| 4-3 | Interface Assignment of the Front Connector       | 4-9 |

**NOTE**

**These instructions do not cover all details or variations in equipment or provide for every circumstance that can arise with installation, operation, or maintenance. If you want further information or if particular problems arise that are not covered sufficiently for your purposes, contact your local Siemens sales office.**

**The contents of this instruction manual shall not become part of or modify any prior or existing agreement, commitment, or relationship. The sales contract contains the entire obligation of Siemens. The warranty contained in the contract between the parties is the sole warranty of Siemens. Any statements contained herein do not create new warranties or modify the existing warranty.**

# Chapter 1

## Technical Description

CPU 946/947 is the standard central processing unit for the S5-155U programmable controller. This chapter explains its application and design and shows how it fits into the S5-155U system structure. This chapter also includes the memory assignment of CPU 946/947 and a list of technical data.

### 1.1 Application

CPU 946/947 is used in the central controller of the S5-155U programmable controller. You can use the 355 memory module to extend the CPU's memory up to 896 Kbytes. This memory capacity enables it to process extensive programs.

CPU 946/947 executes all STEP 5 operations at very high speed and is equipped with floating-point arithmetic. It has an integrated 64 Kword<sup>1</sup> RAM with parity monitoring and a real-time clock.

The following program processing levels are available individually or in combination:

- Cyclic
- Time driven (9 different time bases)
- Interrupt driven (8 interrupts at block boundaries or 4 hardware interrupts)  
The interrupt priorities can be selected. CPU 946/947 can process a user program for communication with communications processors (CPs) in the smooth "STOP" mode.

Using multiprocessing, you can also operate CPU 946/947 with an additional CPU 946/947 or with a CPU of the S5-135U.

CPU 946/947 uses the STEP 5 programming language.

For details on programming see the Programming Guide for the CPU 946/947, CPU 946R/947R in this manual.

### 1.2 Design

CPU 946/947 consists of two plug-in central processing units, 6ES5 946-3UA21/22/23 (CPU 946) and 6ES5 947-3UA21/22/23 (CPU 947).

---

<sup>1</sup> Kword stands for kiloword (one word equals 16 bits). One kiloword equals 1024 words (2<sup>10</sup> words).

## Technical Description

The electronics of CPU 946 are on two printed circuit boards. Two 64-pin male connectors on the mother board link the CPU to the S5 bus and S5 local bus for CPU 946/947 communication to the backplane housing. The front panel is 2<sup>2</sup>/<sub>3</sub> standard plug-in stations wide. Switches, buttons, and LEDs for operating the programmable controller and displaying modes of operation and faults are on the front panel.

The electronics of CPU 947 are on one printed circuit board. Two 64-pin male connectors link the CPU to the S5 local bus for CPU 946/947 communication. It has a 15-pin front connector that enables you to connect the CPU 946/947 to a programmer, operator panel or SINEC interface module.

### 1.3 System Structure

This section shows how CPU 946/947 is connected to the S5 bus and explains the structure of the CPU.

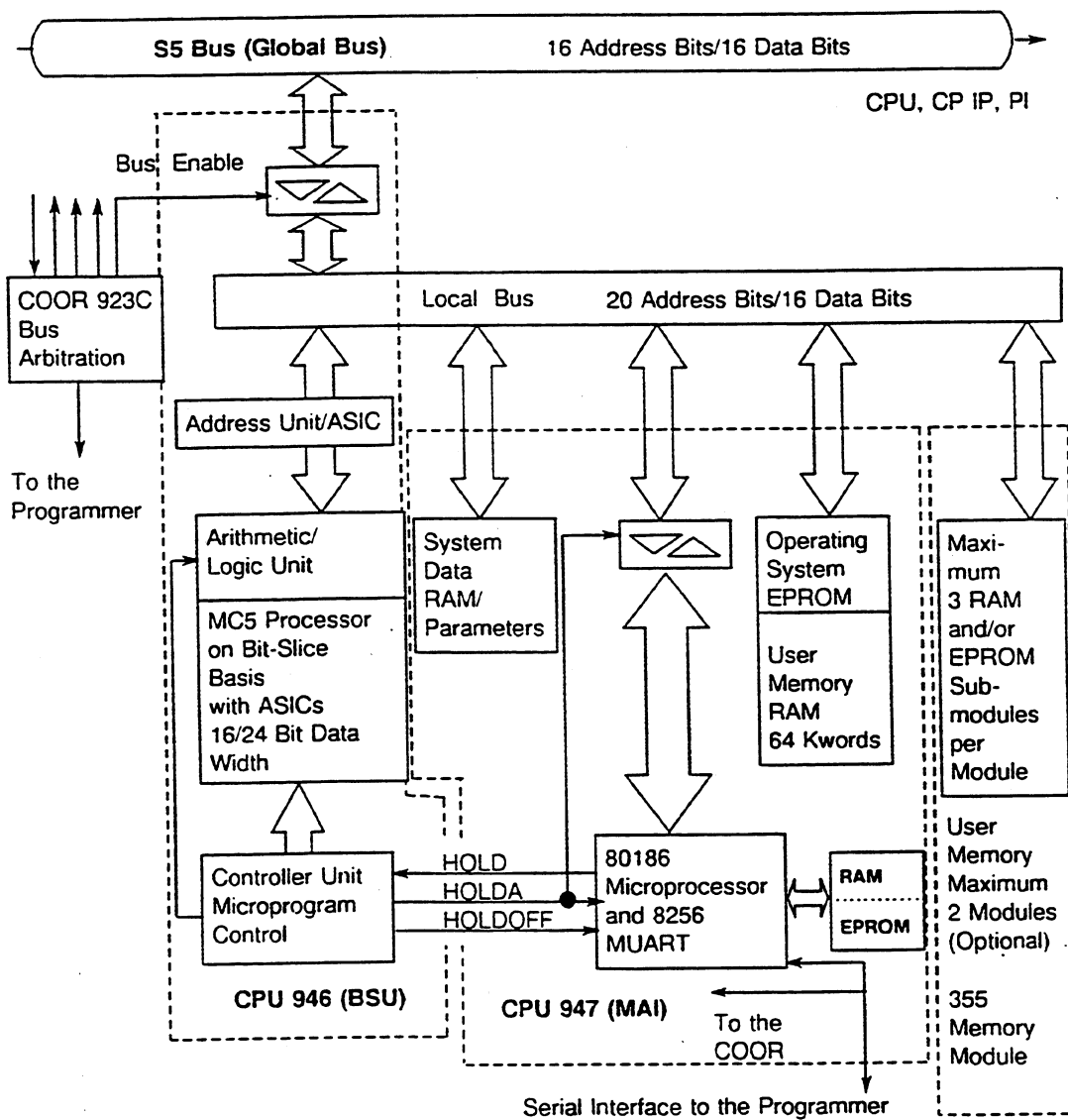


Figure 1-1 Block Diagram of CPU 946/947 with optional 355 Memory Module

The block diagram of CPU 946/947 (Fig. 1-1) shows the following two bus systems:

- **S5 bus (global bus)** - address bus (ADB)           16 bits wide  
                              - data bus (DB)               16 bits wide
  
- **S5 local bus**           - local address bus (LAB)   20 bits wide  
                              - local data bus (LDB)     16 bits wide

The bus drivers between the S5 local bus and the S5 bus are in CPU 946. CPU 947 uses these drivers to access the S5 bus if the local bus enables have been acknowledged.

CPU 946/947 can always access the S5 bus in the single processor mode. With multiprocessing, a CPU can access the bus only through a time-slice procedure. Coordinator 923C uses time-division multiplexing to allocate bus access for each CPU.

The hardware structure of CPU 946/947 is divided into the following two large function areas:

- Bit Slice Unit (**BSU**) - in CPU 946  
  and
- Memory and Interface (**MAI**) - in CPU 947

CPU 946 is divided into the following three main areas:

- control unit
- arithmetic/logic unit
- address unit

CPU 947 is divided into the following two areas:

- memory unit
- communication unit

### **Control Unit**

This unit controls the arithmetic/logic unit, the address unit, and the data flow on the buses. It reads the MC5 operations and executes them in combination with the other units. It reads all interrupts and prioritizes them.

The control unit also updates the clock and timers.

### Arithmetic/Logic Unit

A special logic unit implemented as an applications-specific integrated circuit (ASIC) carries out bit operations.

An arithmetic unit (registered arithmetic/logic unit, or RALU) implemented with bit-slice components (16/24 bits) processes arithmetic and other word operations.

### Address Unit

An address unit implemented as an ASIC provides addresses (20 bits). It provides optimal support for the addressing format of the STEP 5 programming language.

### Memory Unit

The memory unit has a one Mword<sup>1</sup> memory space divided into 16 pages of 64 Kwords each. This division is not significant for the user program assignment. The pages are assigned as follows:

- Page 0 contains the 64 Kword user RAM for code and data. It is permanently integrated in CPU 947.
- Pages 1 to 6 contain RAM memory and pages 8 to D EPROM memory for user programs and user data on the optional plug-in 355 memory module with submodules.
- Page E (system page) contains a 32 Kword EPROM for the operating system. The EPROM is physically in CPU 947. The E page also contains a 31.5 Kword RAM for system data (SD RAM).
- Page F (peripherals page) contains the memory area for the peripheral units. These include inputs/outputs (I/Os), intelligent inputs/outputs (IPs), and communications processors (CPs). These units are physically on the S5 bus.

### Communication Unit

An 80186 microprocessor with its accompanying memories (RAM and EPROM) is the essential component of the communication unit. The communication unit uses the serial interface in the 8256 multifunction universal asynchronous receiver transmitter (MUART) along with its internal timers and input/output ports. The 80186 microprocessor uses its own special bus-lock logic to access the S5 bus.

The communication unit can access the entire memory of CPU 946/947. You can use the interface to the programmer either directly via the front connector of CPU 947 or via the programmer multiplexer (PG-MUX).

---

<sup>1</sup> See Table 1-1

### 1.4 Memory Assignment

The following table shows the memory assignment for CPU 946/947. Pages 1 to 6 and 8 to D represent the user memory space in the optional 355 memory module with memory submodules.

| "Page" | 15                         | 0        |
|--------|----------------------------|----------|
| 0      | 0x2 <sup>10</sup>          | 0 0000 H |
|        | User RAM *)                |          |
| 1      | 64x2 <sup>10</sup>         | 1 0000 H |
|        | User RAM                   |          |
| 2      | 128x2 <sup>10</sup>        | 2 0000 H |
|        | User RAM                   |          |
| 3      | 192x2 <sup>10</sup>        | 3 0000 H |
|        | User RAM                   |          |
| 4      | 256x2 <sup>10</sup>        | 4 0000 H |
|        | User RAM                   |          |
| 5      | 320x2 <sup>10</sup>        | 5 0000 H |
|        | User RAM                   |          |
| 6      | 384x2 <sup>10</sup>        | 6 0000 H |
|        | User RAM                   |          |
| 7      | 448x2 <sup>10</sup>        | 7 0000 H |
|        | not assigned               |          |
| 8      | 512x2 <sup>10</sup>        | 8 0000 H |
|        | User EPROM                 |          |
| 9      | 576x2 <sup>10</sup>        | 9 0000 H |
|        | User EPROM                 |          |
| A      | 640x2 <sup>10</sup>        | A 0000 H |
|        | User EPROM                 |          |
| B      | 704x2 <sup>10</sup>        | B 0000 H |
|        | User EPROM                 |          |
| C      | 768x2 <sup>10</sup>        | C 0000 H |
|        | User EPROM                 |          |
| D      | 832x2 <sup>10</sup>        | D 0000 H |
|        | User EPROM                 |          |
| E      | 896x2 <sup>10</sup>        | E 0000 H |
|        | System Page*)              |          |
| F      | 960x2 <sup>10</sup>        | F 0000 H |
|        | S5 Bus<br>Peripherals Page |          |
|        | 1024x2 <sup>10</sup> -1    | F FFFF H |

Maximum Configuration with RAM Submodules (No EPROM Configuration)

Maximum Configuration with EPROM Submodules (No RAM Configuration)

\*) Memory in CPU 947

Table 1-1 CPU 946/947 Memory Assignment

## 1.5 Technical Specifications

|   |  |                   |              |
|---|--|-------------------|--------------|
| Degree of protection                                    | IP <sup>1</sup> 00   |                   |              |
| Permissible ambient temperature                         | 0° C to +55° C   |                   |              |
| - operation   | - 40° C to +70° C  |                   |              |
| - storage and transport                                 |  |                   |              |
| Relative humidity                                       | max. 95% at 25° C, noncondensing   |                   |              |
| Operating altitude                                      | Up to 3500 m (11,483.5 ft.) <sup>2</sup> above sea level                             |                   |              |
| Supply voltage  | 5 V DC, ± 5%   |                   |              |
| Current consumption                                     |  |                   |              |
| - CPU 946   | approx. 6 A  |                   |              |
| - CPU 947   | approx. 2 A  |                   |              |
| Back-up current CPU 946/947                             | approx. 10 µA  |                   |              |
|   | <b>P Area</b>  | <b>O (Q) Area</b> | <b>Total</b> |
| Digital inputs  |  |                   |              |
| - with process image                                    | max. 1024  | -                 | max. 1024    |
| - without process image                                 | max. 1024  | max. 2048         | max. 3072    |
| Analog inputs   | max. 64  | max. 128          | max. 192     |
| Digital outputs   |  |                   |              |
| - with process image                                    | max. 1024  | -                 | max. 1024    |
| - without process image                                 | max. 1024  | max. 2048         | max. 3072    |
| Analog outputs  | max. 64  | max. 128          | max. 192     |
| Flags   | 2048   |                   |              |
| S Flags   | 32768  |                   |              |
| Timers  | 256  |                   |              |
| Counters  | 256  |                   |              |
| Size of the user memory                                 | 64 Kwords (16 bits wide)   |                   |              |
| Memory extension with the 6ES5 355-3UA.. and submodules | max. 384 Kwords (16 bits wide)   |                   |              |
| Maximum memory configuration                            | 448 Kwords (16 bits wide) (complete with RAM, or 768 Kbyte EPROM plus 128 Kbyte RAM) |                   |              |
| Program blocks (PB)                                     | 256  |                   |              |
| Sequence blocks (SB)                                    | 256  |                   |              |
| Function blocks (FB)                                    | 256  |                   |              |
| Data blocks (DB)  | 256 (254 for user programs)  |                   |              |
| Extended function blocks (FX)                           | 256  |                   |              |
| Extended data blocks (DX)                               | 256 (255 for user programs)  |                   |              |

<sup>1</sup> IP stands for an environmental rating. The first number (0 to 6) after IP is the dry particle rating and the second number (0 to 8) is the moisture rating.

<sup>2</sup> Where dimensions are indicated in meters and feet, the conversion factor used is 3.281 (1 m = 3.281 ft.) with feet rounded off to the nearest tenth of a foot.



|   |   |
|---|---|
| Machine code  | MC5, code of the STEP 5 programming language                                  |
| Transmission speed of the programmer serial interface | 9600 bits/sec.  |
| Scan monitoring time                                  | Can be set using software. Default setting is 200 msec.                       |
| Acknowledge monitoring time                           | 150 $\mu$ sec.  |
| Dimensions (w x h x d)                                |   |
| - CPU 946   | 40.64 mm x 233.4 mm x 160 mm<br>(1.58 in. x 9.10 in. x 6.24 in.) <sup>1</sup> |
| - CPU 947   | 20.32mm x 233.4mm x 160mm<br>(0.79in. x 9.10in. x 6.24in.)                    |
| Weight  |   |
| - CPU 946   | approx. 900 g (1.98 lbs.) <sup>2</sup>  |
| - CPU 947   | approx. 500 g (1.10 lbs.)   |

<sup>1</sup> Where dimensions are indicated in millimeters and inches, the conversion factor used is 0.039 (1 mm = 0.039 in.) with inches rounded off to the nearest hundredth of an inch.

<sup>2</sup> Where weights are indicated in kilograms and pounds, the conversion factor used is 2.2 (1 kg = 2.2 lbs.) with pounds rounded off to the nearest hundredth of a pound.

## **1.6 Functional Capabilities**

CPU 946/947 processes, with a few exceptions, the complete range of STEP 5 operations. The operations are listed and described in the *S5-155U List of Operations* and in the *S5-155U: STEP 5 Programming Guide*.

The programming guide also describes the functions of the operating system, the user interface to the operating system, and the modes of operation.

Interrupt driven processing has the following two modes:

- 150S controller mode - Input byte 0 (IB0) is scanned at block boundaries. When a signal changes, program processing branches to the assigned OB.
- 155U controller mode - Program processing branches to the assigned OB at operation boundaries (at the end of each STEP 5 operation statement) when an interrupt occurs at one of the four interrupt inputs of CPU 946/947.

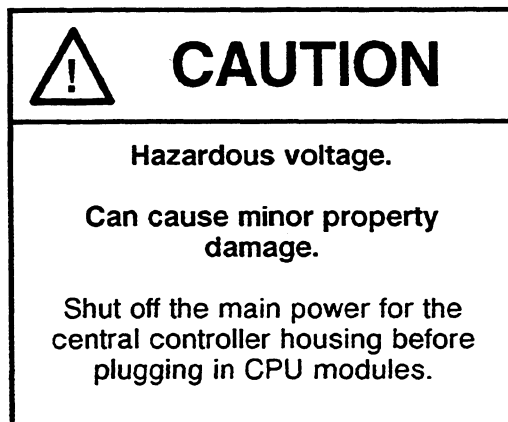
See section 5.5 of the *S5-155U Instructions* and the *S5-155U Central Controller Housing Hardware and Installation Guide* for information on interrupts and interrupt I/O modules (e.g., the 432 digital input module).

## Chapter 2 Installation and Operator Information

This chapter tells you how to install and remove CPU 946/947. It also explains the control switches and LEDs on the front panel of CPU 946/947.

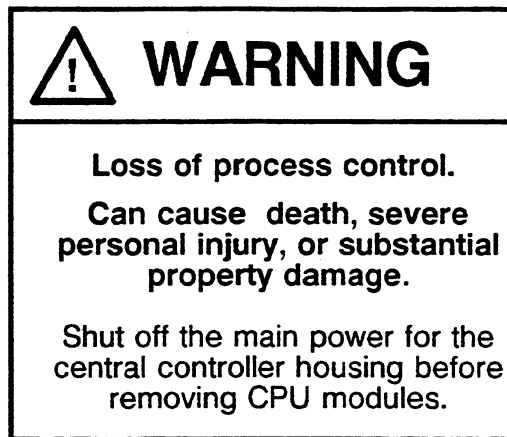
### 2.1 Plugging In and Removing Modules

Like all modules of the S5-155U programmable controller, CPU 946/947 is easily plugged into or removed from the central controller housing.



To plug CPU 946/947 into the central controller housing, use the following procedure:

- Select the correct slot according to the printed strip on the locking bar at the top of the housing.
- Insert the module so that it is aligned with the tracks at the bottom and top of the slot, being careful not to skew it. Push it all the way back until it engages with the connector and the release lever snaps into place.
- Tighten the plastic screw at the bottom of the module with a screwdriver.
- Secure the locking bar at the top of the module.



To remove CPU 946/947 from the central controller housing, use the following procedure:

- Loosen the locking bar at the top of the central controller housing.
- Loosen the locking screw at the bottom of the module.
- Push the release lever and pull the module towards you.

Do not dismantle the CPU 946 double-width module.

## **2.2 Control Switches and LEDs**

CPU 946/947 has control switches and LEDs on its front panel for operator control and information, these are explained in the following sections.

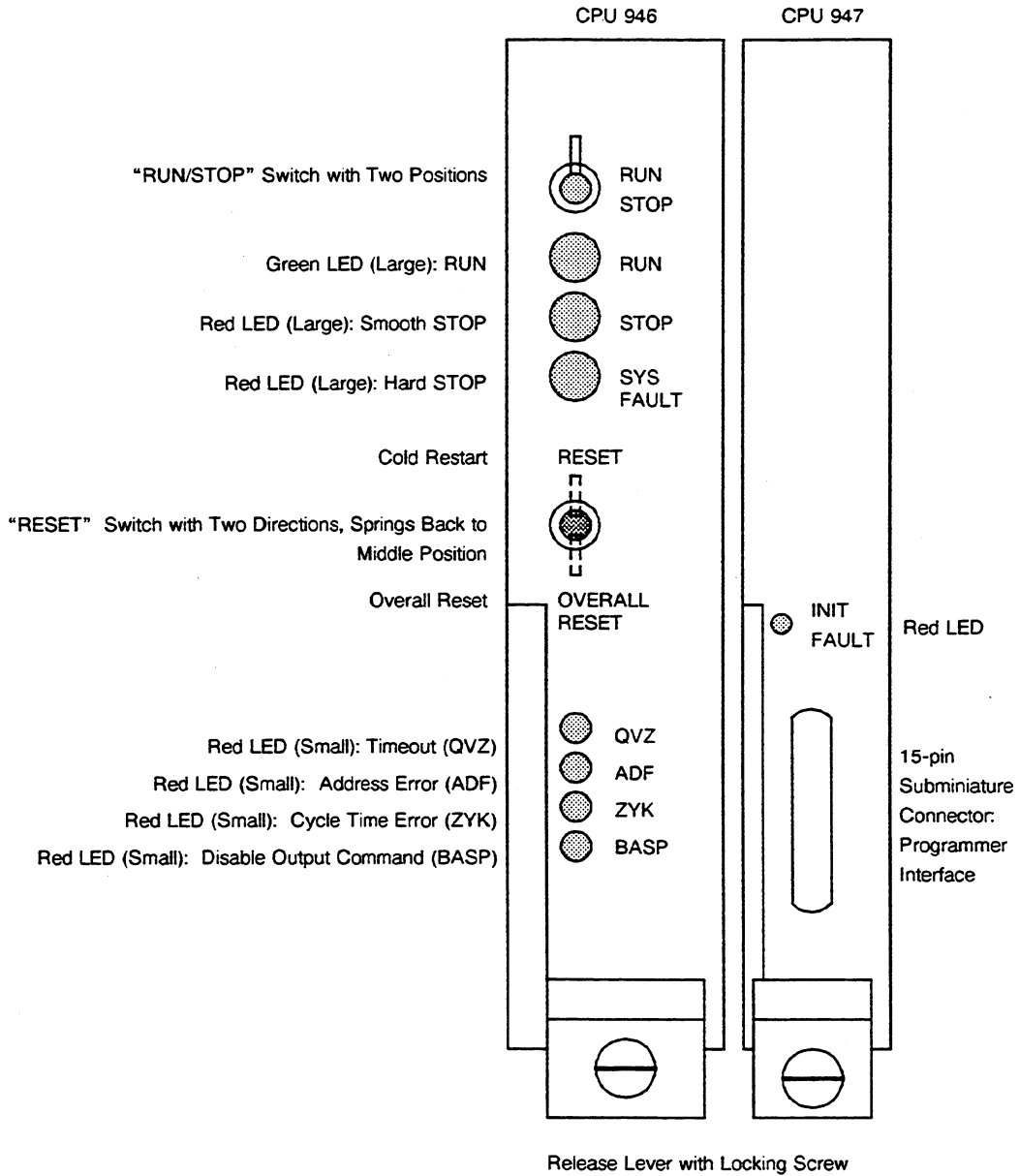


Figure 2-1 Front Panel of CPU 946/947

| Switch Settings | Switch Position | Result of Switching from "STOP" to "RUN" |
|-----------------|-----------------|--|
| RESET           | Top             | Cold restart                             |
| 0               | Middle          | Warm restart                             |
| OVERALL RESET   | Bottom          | Overall reset                            |

Table 2-1 CPU 946/947 "RESET" Switch Settings

For more information on the modes of operation see the *S5-155U: STEP 5 Programming Guide* for CPU 946/947, CPU 946R/947R in this manual.

### 2.2.1 "RUN/STOP" Switch

This subsection describes the function of the "RUN/STOP" switch on the front panel of CPU 946/947.

#### "RUN" Position

When the "RUN/STOP" switch is in the "RUN" position, CPU 946/947 is in cyclic operation if the green "RUN" LED is lit at the same time. During cyclic operation, CPU 946/947 goes through the following continuous cycle: it reads in the process image of the inputs, processes the user program according to the call sequence in organization block 1 (OB1), transfers the process image to the outputs, updates the communication flags (as programmed in DB 1) and triggers scan time monitoring.

Timer update and time and process interrupts briefly interrupt cyclic processing. Acknowledged faults and online programmer functions can also interrupt this processing. The following errors in the system, in the programmable controller unit, and in the program can interrupt cyclic processing:

- power failure (NAU)
- timeout (QVZ)
- substitution error (SUF)

You can program reactions to these errors in organization blocks.

With the "RUN/STOP" switch in the "RUN" position at power up, an automatic warm restart takes place if the programmable controller was in cyclic operation before the power was shut off and if no error entries other than "NAU" are stored.

#### "STOP" Position

After you switch CPU 946/947 from "RUN" to "STOP," it goes into the smooth "STOP" mode. The CPU outputs the "BASP" signal, disabling the digital output modules. At this time, you can use the programmer to execute the FORCE function. The processor then suppresses the "BASP" signal and enables the digital outputs.

### 2.2.2 "RESET" Switch

This subsection describes the function of the "RESET" switch on the front panel of CPU 946/947 in relation to its two possible positions "OVERALL RESET" and "RESET" (see also Table 2-1).

#### Overall Reset Function

An overall reset erases and initializes all RAMs. Afterward, the "STOP" LED stays lit. Perform an overall reset as follows:

- Hold the "RESET" switch in the "OVERALL RESET" position.
- Move the "RUN/STOP" switch from "STOP" to "RUN" and then to "STOP" again.
- Release the "RESET" switch.

**Result:** The "STOP" LED flashes quickly, requesting an overall reset.

**Note:** At this point, you can interrupt an overall reset by moving the "RUN/STOP" switch from "STOP" to "RUN" and then to "STOP" again **without** touching the "RESET" switch. This stops an overall reset. CPU 946/947 remains in the "STOP" mode. The "STOP" LED stays lit.

- Again hold the "RESET" switch in the "OVERALL RESET" position.
- Move the "RUN/STOP" switch from "STOP" to "RUN" and then to "STOP" again.
- Release the "RESET" switch.

**Result:** An overall reset is executed. CPU 946/947 remains in the "STOP" mode and the "STOP" LED stays lit. Afterward, you can execute only a cold restart.

**Restart Function**

Use the "RESET" switch on CPU 946/947 to perform a cold or warm restart as follows (see chapter 3.1 for more information):

**2.2.3 Operating Status LEDs**

This subsection explains the function of the operating status LEDs on the front panel of CPU 946/947. The table below provides an overview of their respective conditions and meanings.

| LED                                   | Condition       | Meaning                             |
|---------------------------------------|-----------------|-------------------------------------|
| RUN (green)                           | Lit             | Cyclic program processing           |
| STOP (red)<br>(smooth "STOP" mode)    | Lit             | Power on, in "STOP" mode, no errors |
|                                       | Flashes quickly | Overall reset requested             |
|                                       | Flashes slowly  | Cold restart required               |
| SYS FAULT (red)<br>(hard "STOP" mode) | Lit             | System error                        |

**"RUN" LED**

When the green "RUN" LED stays lit, it indicates cyclic program processing.

**"STOP" LED (smooth "STOP" mode)**

The red "STOP" LED can display the following three "STOP" statuses, lit continuously, flashing slowly or flashing quickly. In the smooth STOP mode the CPU 946/947 can cyclically process a user program to communicate with the CPs.

- "STOP" LED stays lit
- This occurs after the power is turned on if the "RUN/STOP" switch is at "STOP" and no errors occurred during initialization. System restart is possible. The following actions or errors can trigger the smooth "STOP" mode in

### Single processor operation:

- Moving the "RUN/STOP" switch from "RUN" to "STOP"
- Executing the "PC STOP" programmer function
- Executing an overall reset
- Programmable controller errors that are not assigned to an individual CPU (BAU, NAU, PEU<sup>1</sup>)
- After completing the programmer function End Program Test at another CPU.

The following actions can trigger the smooth "STOP" mode in

### Multiprocessor operation:

- Moving the "RUN/STOP" switch on Coordinator 923C from "RUN" to "STOP"
- A different CPU or Coordinator 923C causes the smooth "STOP." (The "STOP" LED on a CPU not responsible for the smooth "STOP" stays lit.)
- Executing the "PC STOP" programmer function with a different CPU

- "STOP" LED flashes quickly (approximately twice per second)

This occurs after you or the system program requests an overall reset (see subsection 2.2.2). System restart is possible only if you perform an overall reset or if you eliminate any hardware errors and then perform an overall reset.

- "STOP" LED flashes slowly (approximately once per second)

This occurs under the following conditions to indicate that a cold restart is required:

- An error occurs during cyclic program processing of the CPU. The CPU is in the "STOP" mode because no appropriate error reaction was programmed. Switch the "RUN/STOP" switch to "STOP", the LED is then lit, as long as the error does not occur again.
- An operator error (e.g. an illegal restart mode or DB1/DX0 error).
- A STOP operation (STP and STS)<sup>2</sup> is being processed in the user program.
- In addition to the slowly flashing "STOP" LED, the following LEDs light up when certain programming and programmable controller errors occur:
  - "QVZ" LED - timeout
  - "ADF" LED - addressing error
  - "ZYK" LED - cycle time exceeded
- The "End Program Test" programmer function is running at the CPU. ("Program Test" is a debugging tool. See the *S5-155U: STEP 5 Programming Guide* for more information.)

---

<sup>1</sup> BAU, NAU, and PEU stand for "battery failure," "power failure," and "I/O not operable," respectively.

<sup>2</sup> See the *S5-155U List of Operations* for an explanation of the STP and STS operations.



### **"SYS FAULT" LED (hard "STOP" mode)**

The "SYS FAULT" LED lights up if a system error prevents the system program from operating properly. CPU 946/947 goes into the hard "STOP" mode. This ensures that operation does not continue with a defective system program.

The following conditions can trigger the hard "STOP" mode:

- timeout (QVZ) or parity error (PARE) in the system RAM or EPROM
- interrupt stack (ISTACK) overflow (STUEU)
- STEP 5 operation "stop for time interrupt processing" (STW)

#### **NOTE**

**Only by switching the main power off and then on can you cancel the hard "STOP" mode (CPU 946/947 stopped).**

## **2.2.4 Error and Signal LEDs**

### **"QVZ" LED**

The "QVZ" LED lights up under any of the following conditions:

- if the program attempts to address a peripheral module in single processor operation after a cold restart of CPU 946/947 in the area of the process image (IB 0 through IB 127, QB 0 through QB 127) which was entered in track 9, and the module no longer responds.
- if the program attempts to address a peripheral module in single or multiprocessor operation. The module does not respond, even though it is entered in DB1 (address list) and the module was detected as being plugged in during a cold restart.
- if a peripheral module does not respond or no longer responds when the program attempts to access it directly (e.g., using the L PB, L PW, T PB, T PW, L OB, L OW, T OB, or T OW operation<sup>1</sup>).

The following situations could cause the "QVZ" LED to light up:

- A module has failed.
- A module is removed while the programmable controller is operating, or in "STOP" mode or the power was shut off and no cold restart performed after power was turned on.

You can program an interface reaction to QVZ in organization blocks. (See the *S5-155U: STEP 5 Programming Guide* in this manual for more information.)

---

<sup>1</sup> See the *S5-155U List of Operations* for more information.

### **"ADF" LED**

During cold restart of CPU 946/947, the operating system sets up a ninth track in the control RAM of CPU 947. All available I/O modules are marked in the ninth track in the process image area. During multiprocessing or programming of the address list in DB1, the operating system sets up a ninth track with the help of DB1. A check is made to see if the corresponding addresses are acknowledged. If in the user program an address in the process image is addressed under which no module has been configured, CPU 946/947 interrupts cyclic program processing (default). You can program a reaction to ADF in OBs (see the *S5-155U: STEP 5 Programming Guide*).

### **"ZYK" LED**

The "ZYK" LED lights up if the maximum cycle time is exceeded. The cycle time is the total of the scan times of all user program parts (cyclic plus time controlled, plus process interrupt controlled). The "ZYK" error signal interrupts cyclic program processing. You can program a reaction to ZYK in OBs (see the *S5-155U: STEP 5 Programming Guide*).

### **"BASP" LED**

The "BASP" LED lights up if command output is disabled. Digital outputs are switched directly to an "OFF" status. The "BASP" signal does not reset the memory registers in digital input/output modules. The "BASP" signal is output when the power supply unit is switched on and off, when the voltage is low, or when the CPU is in the "STOP" mode.

### **"INIT FAULT" LED**

During system restart, the orange "INIT FAULT" LED on CPU 947 stays lit and then goes out after the restart procedure is completed. If an error prevents completion of the restart procedure, this LED flashes.

## 2.3 Restart Procedure for the CPU 946/947

You can start up the CPU 946/947 (together with the memory extension using the 355 memory module with memory submodules) without a programmer.

Ensure that the modules are plugged into the correct slots on the central controller 155U.

The back-up battery in the central controller 155U must be positioned in its slide-in module and fully functional before the CPU can go into operation.

Proceed as follows:

- 1) Switch the "RUN/STOP" switch on the CPU 946 to "STOP".
- 2) Switch on the power supply
  - green LED "5 V DC power supply ok" on the power supply illuminates
  - green LED "15 V/24 V DC power supply ok" on the power supply illuminates
  - red LED "STOP" on the CPU 946 illuminates
  - small red LED "BASP" on the CPU 946 illuminates.
- 3) Hold the switch in the position "OVERALL RESET" and simultaneously switch the "RUN/STOP" switch to "RUN":
  - red LED "STOP" on the CPU 946 flashes quickly.
- 4) Repeat procedure 3):
  - red LED "STOP" on the CPU 946 lights continuously.

The overall reset is then complete.

If, in addition, the red LED "SYS FAULT" on the CPU 946 should illuminate, an error has occurred during the overall reset. In that case repeat the overall reset according to 3) and 4), or switch off the power supply and begin again with 1) before employing other measures (reading out ISTACK with the programmer, exchanging the CPU etc.).

- 5) Switch the "RUN/STOP" switch to "STOP".
- 6) Hold the switch in the position "RESET" and switch the "RUN/STOP" switch to "RUN":
  - the red LED "STOP" goes out and the orange LED "INIT FAULT" illuminatesand after a short interval:
  - the orange LED "INIT FAULT" goes out
  - the green LED "RUN" illuminates
  - the small red LED "BASP" goes out.

The CPU is now in cyclic mode, but without a user program.

While the CPU runs up, various tests are carried out. If errors occur during restart, this is indicated by the orange LED "INIT FAULT" flashing.

For maintenance and service purposes or in case of a fault, an initial report as to whether the CPU or the system program are still functioning at all can be obtained using the described restart procedure or user program. If the 355 memory module and RAM module are also plugged during a test of this kind, their functions are included in the test.

## **2.4 Operation of Peripheral Modules**

The hardware of the CPU 946/947 works solely with addresses that are 20 bits wide. Consequently when you work with absolute addressing, this hardware is not compatible with the S5-150U programmable controller (see *STEP 5 Programming Guide*) for CPU 946/947, CPU 946R/947R in this manual.

The S5 bus (global bus) has addresses that are 16 bits wide, as does the S5-155U. The operation of digital and analog input/output modules in the P area and O (Q) area, the operation of communications processors (CPs) with page frame addressing and the use of intelligent input/output modules (IPs) all function in the same way as for the S5-150U and S5-135U.

## Chapter 3 Operation

This chapter defines the types of restart for CPU 946/947 and explains the restart procedure and describes the programmer interfaces.

### 3.1 Restart Types

When CPU 946/947 starts up, its operating system determines and sets up the data necessary for cyclic operation. Afterwards, cyclic program processing begins (see the Programming Guide). The system program differentiates between the 3 types of restart described below:

#### 3.1.1 Cold Restart

Flag, timer, and counter data and the I/O process images are cleared. User program processing starts over again.

The programmable controller must be in the "STOP" mode. With multiprocessing, the "RUN/STOP" switch on Coordinator 923C must be in the "STOP" position. Reset CPU 946/947 and put it into cyclic program processing using the following procedure:

- Hold the CPU 946/947 "RESET" switch in the "RESET" position.
- Move the "RUN/STOP" switch from "STOP" to "RUN."
- With multiprocessing, move the "RUN/STOP" switch on all CPUs in the S5-155U from "STOP" to "RUN." Then move the "RUN/STOP" switch on Coordinator 923C from "STOP" to "RUN."

#### 3.1.2 Manual Warm Restart

Flag, timer, and counter data and the I/O process images are maintained. User program processing resumes from the point at which it was interrupted.

For a warm restart to function, the programmable controller must have been in cyclic operation before it went into the "STOP" mode. With multiprocessing, the "RUN/STOP" switch on Coordinator 923C must be in the "STOP" position. Put the CPU into cyclic program processing using the following procedure:

- Leave the CPU 946/947 "RESET" switch in the middle position.
- Move the "RUN/STOP" switch from "STOP" to "RUN."
- With multiprocessing, move the "RUN/STOP" switch on Coordinator 923C from "STOP" to "RUN."

### 3.1.3 Automatic Warm Restart

Flag, timer, and counter data and the I/O process images are maintained. User program processing resumes from the point at which it was interrupted.

An automatic warm restart is carried out after switching on the power supply under the following conditions:

- the PLC was in cyclic operation before the power was switched off or cut off,
- the "RUN/STOP" switch on the CPU 946/947 is still in the position "RUN" (in multiprocessing, also on the other CPUs and the 923 coordinator),
- user memory submodules were not removed or replaced; new ones were not inserted, and
- the back-up battery is functioning properly (i.e. the data in the RAM has been retained).

### 3.1.4 Automatic Cold Restart

You can select this restart mode instead of an automatic warm restart by programming DX 0 accordingly.

### 3.1.5 Test Mode

You will find information on the Test mode in the Programming Guide.

## 3.2 Programmer Interface

You can use the programmer interface on CPU 947 either via the front connector or Coordinator 923C.

### Note:

- 1) It is not possible to operate the programmer interface via the front connector of CPU 947 and Coordinator 923C simultaneously. Just switching the programmer online, even without a command, operates the interface. Electrically, there is only one programmer interface. You can operate this single interface via two separate connections.
- 2) If you wish to operate a Coordinator 923C in the 155U controller mode (selectable in extended data block DX0), the S5-155U is automatically in multiprocessor operation. You must load DB1 with the address list: the CPU does not enter the "RUN" mode without DB1.
- 3) Do not plug in a coordinator if the CPU is in the 150U controller mode (default setting or selectable in extended data block DX0). Otherwise the CPU enters the "STOP" mode. You cannot use the PG-MUX of the coordinator in the 150U controller mode.

You can establish a connection to a programmer in any operating mode of the CPU.

## Chapter 4 Maintenance

This chapter explains the central register for error addresses of CPU 946/947. It also shows the layout of the jumpers on the CPU and describes the interface assignment for its backplane connectors and front connector.

### 4.1 Central Register for Error Addresses

CPU 947 or the 355 memory module can trigger a parity interrupt. When this happens, the first incorrect address is stored in the error register in CPU 947.

If a timeout signal (QVZ) is detected when CPU 947 tries to access the local bus, the first QVZ address is stored in this register instead of the parity error address.

Read addresses E8004H and E8000H to determine the 20-bit wide error addresses on the local bus (e.g., using the programmer PC INFO function "OUTP ADDR"). Make sure you output the error addresses under E8004H first and then under E8000H.

Reading out the register resets any parity interrupts in the queue from CPU 947 or from the memory module. New error addresses cannot be stored until this resetting takes place. The readout does not erase the contents of the register. It is maintained until a new error address is stored.

Format of an Error Address Register:

E8004H: High part of the error address

Data bit 0 = Address bit 16 / Data bit 1 = Address bit 17

Data bit 2 = Address bit 18 / Data bit 3 = Address bit 19

The remaining data bits are insignificant.

E8000H: Low part of the error address

Data bits 0 to 15 correspond to error address bit numbers 0 to 15.

## 4.2 Jumper Layout

The following figures show the jumper layout on CPU 946/947.

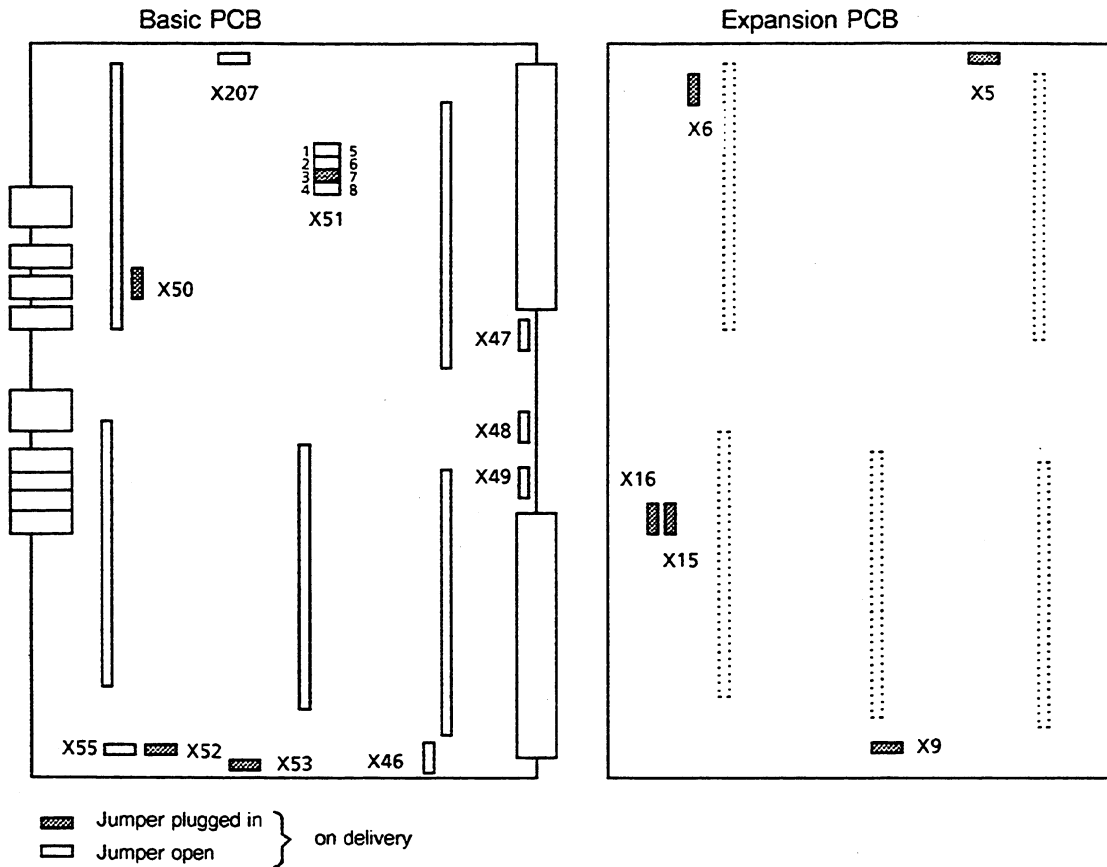


Figure 4-1 Jumper Layout on CPU 946 (6ES5 946-3UA21)



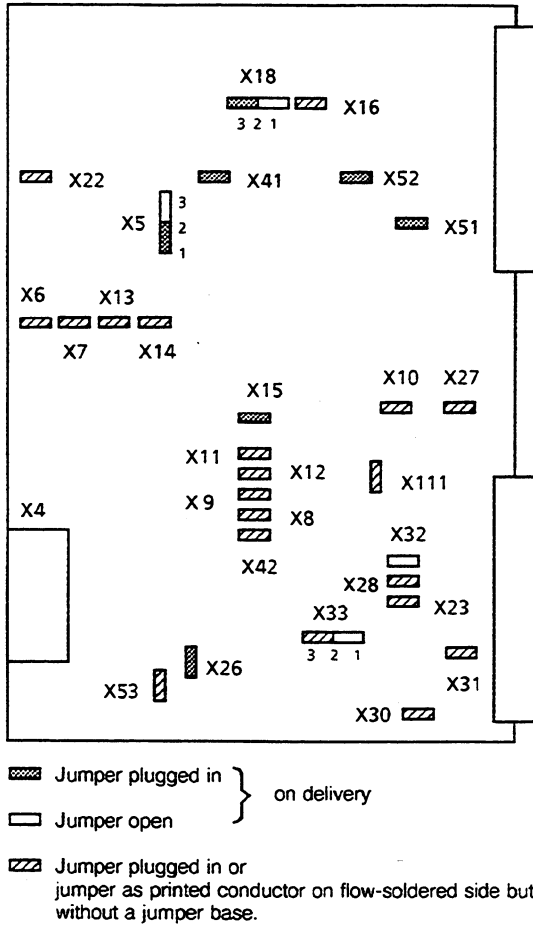


Figure 4-2 Jumper Layout on CPU 947 (6ES5 947-3UA21)

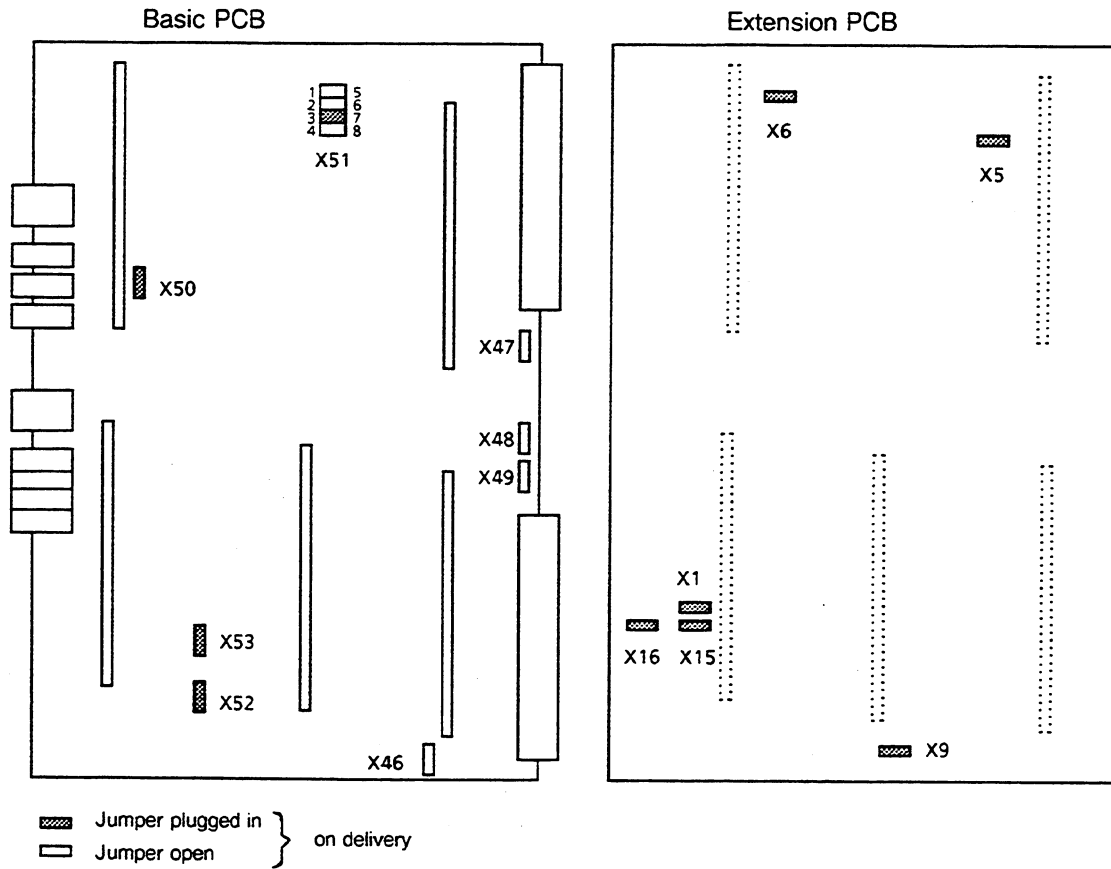


Figure 4-3 Jumper Layout on CPU 946 (6ES5 946-3UA22)

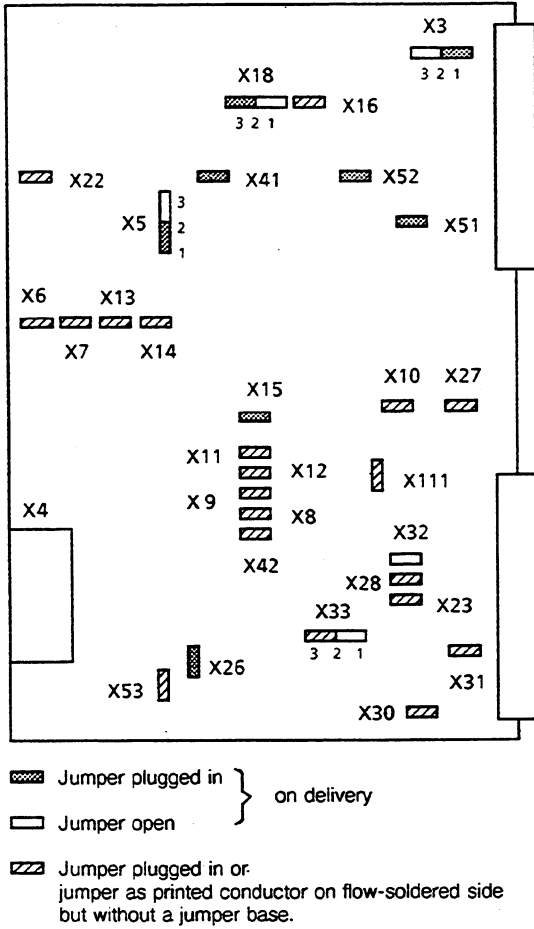


Figure 4-4 Jumper Layout on CPU 947 (6ES5 947-3UA22)

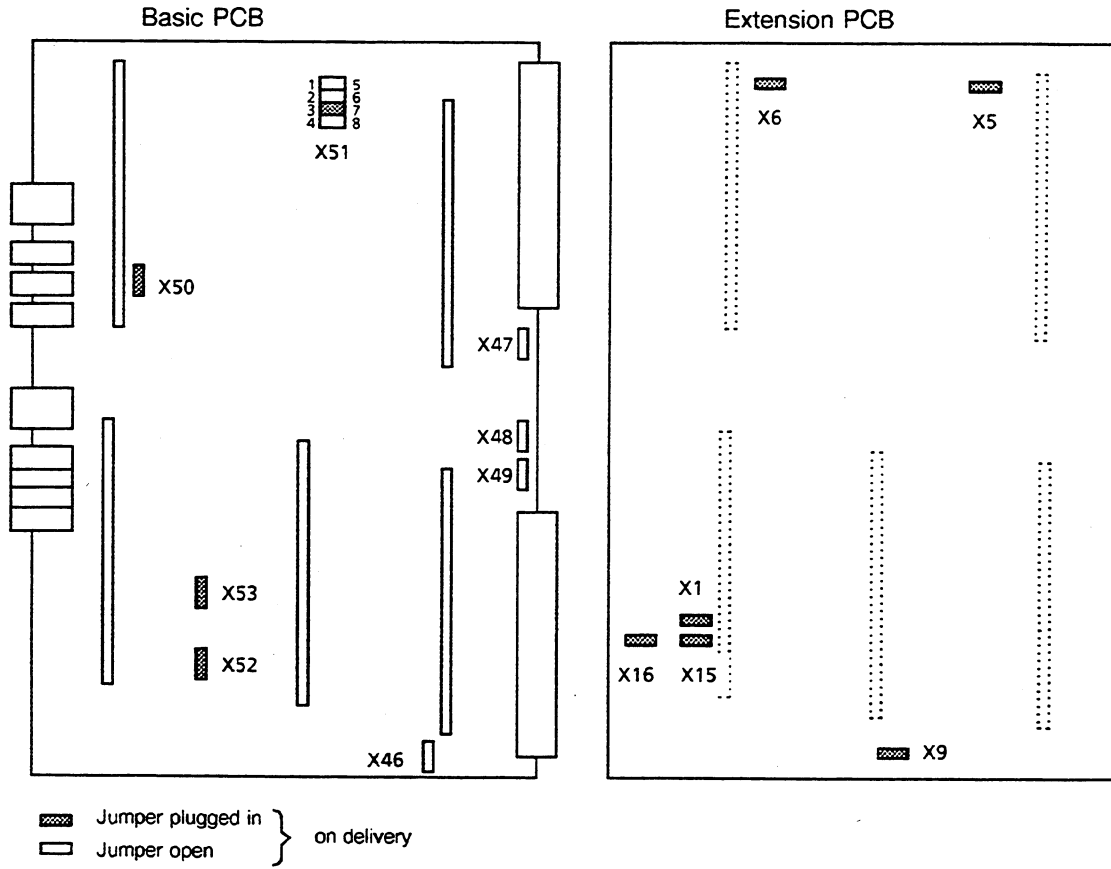


Figure 4-5 Jumper Layout on CPU 946 (6ES5 946-3UA23)

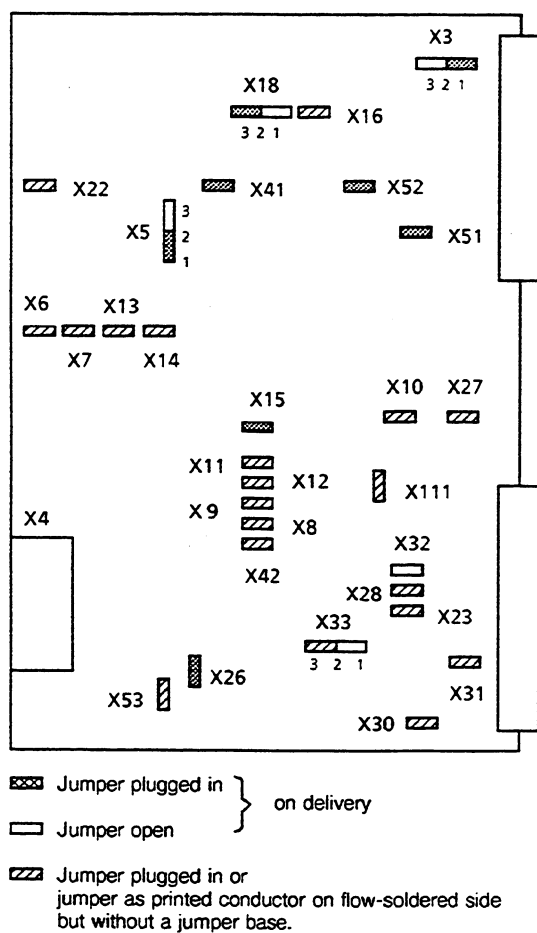


Figure 4-6 Jumper Layout on CPU 947 (6ES5 947-3UA23)

#### Note

**Do not close jumpers X46 to X49 when the CPU is in 150U controller mode.**

You cannot change the jumpers on the CPU 946/947 - the only exception being X46 to X49. By closing these jumpers on the basic printed circuit board you can assign CPU 946/947 the interrupt signals that come from modules with interrupt generation as follows (see the *S5-155U Central Controller Hardware and Installation Guide* and *CPU 946/947 Programming Guide*):

- X46 → INT G
- X47 → INT F
- X48 → INT E
- X49 → INT A/B (depending on location)

### 4.3 Interface Assignment of the Backplane Connectors

| CPU 946 |       |         |         |        | CPU 947 |       |         |         |        |
|---------|-------|---------|---------|--------|---------|-------|---------|---------|--------|
|         | z     | b       | d       | f      |         | z     | b       | d       | f      |
| 2       | + 5 V | M (5 V) | M (5 V) | LAB 0  | 2       | + 5 V | M (5 V) | M (5 V) | LAB 0  |
| 4       | + 5 V | PESP    | UBAT    | LAB 1  | 4       | + 5 V |         | UBAT    | LAB 1  |
| 6       | RESET | ADB 0   | ADB 12  | LAB 2  | 6       | RESET |         |         | LAB 2  |
| 8       | MEMR  | ADB 1   | ADB 13  | LAB 3  | 8       |       |         |         | LAB 3  |
| 10      | MEMW  | ADB 2   | ADB 14  | LAB 4  | 10      |       |         |         | LAB 4  |
| 12      | RDY   | ADB 3   | ADB 15  | LAB 5  | 12      |       |         |         | LAB 5  |
| 14      | DB 0  | ADB 4   | IRx     | LAB 6  | 14      |       |         | IR      | LAB 6  |
| 16      | DB 1  | ADB 5   | M (5 V) | LAB 7  | 16      |       |         | M (5 V) | LAB 7  |
| 18      | DB 2  | ADB 6   | INTMC   | LAB 8  | 18      |       |         | INTMC   | LAB 8  |
| 20      | DB 3  | ADB 7   | INTAS   | LAB 9  | 20      |       |         | INTAS   | LAB 9  |
| 22      | DB 4  | ADB 8   | IRE     | LAB 10 | 22      |       |         |         | LAB 10 |
| 24      | DB 5  | ADB 9   | IRF     | LAB 11 | 24      |       |         |         | LAB 11 |
| 26      | DB 6  | ADB 10  | IRG     | LAB 12 | 26      |       |         |         | LAB 12 |
| 28      | DB 7  | ADB 11  | DSI     | LAB 13 | 28      |       |         | DSI     | LAB 13 |
| 30      |       | BASP    | BUSEN   | LAB 14 | 30      |       |         | BUSEN   | LAB 14 |
| 32      | HALT  | M (5 V) | BASPq   | LAB 15 | 32      | HALT  | M (5 V) | BASPq   | LAB 15 |

Table 4-1 Interface Assignment of the Backplane Connector 1

|    | z        | b        | d        | f      |    | z        | b        | d        | f      |
|----|----------|----------|----------|--------|----|----------|----------|----------|--------|
| 2  | + 5 V    | M (5 V)  | LAB 16   | LDB 0  | 2  | + 5 V    | M (5 V)  | LAB 16   | LDB 0  |
| 4  | DB 12    | DB 8     | LAB 17   | LDB 1  | 4  |          |          | LAB 17   | LDB 1  |
| 6  | DB 13    | DB 9     | M (5 V)  | LDB 2  | 6  |          |          | M (5 V)  | LDB 2  |
| 8  | DB 14    | DB 10    | LAB 18   | LDB 3  | 8  |          |          | LAB 18   | LDB 3  |
| 10 | DB 15    | DB 11    | LAB 19   | LDB 4  | 10 |          |          | LAB 19   | LDB 4  |
| 12 | M (5 V)  | TAU      | LMEMR    | LDB 5  | 12 | M (5 V)  | TAU      | LMEMR    | LDB 5  |
| 14 | NAU      | PG-BUSX  | LMEMW    | LDB 6  | 14 | NAU      |          | LMEMW    | LDB 6  |
| 16 | BAU      | PG-BUSY  | LRDY     | LDB 7  | 16 | BAU      |          | LRDY     | LDB 7  |
| 18 | + 5 V    | M (5 V)  | NS       | LDB 8  | 18 | + 5 V    | M (5 V)  | NS       | LDB 8  |
| 20 | HOLD     | STEU     | HOLDA    | LDB 9  | 20 | HOLD     | STEU     | HOLDA    | LDB 9  |
| 22 | PEU      | STOPPA   | (TxD)    | LDB 10 | 22 | PEU      | STOPPA   | TxD      | LDB 10 |
| 24 | GEP      | M (5 V)  | HOLDO    | LDB 11 | 24 | GEP      | M (5 V)  | HOLDO    | LDB 11 |
| 26 | PARE     | (RxD)    | TEST     | LDB 12 | 26 | PARE     | RxD      | TEST     | LDB 12 |
| 28 | + 5 V    | PERO     |          | LDB 13 | 28 | + 5 V    | PERO     |          | LDB 13 |
| 30 | M (24 V) | M (24 V) | M (24 V) | LDB 14 | 30 | M (24 V) | M (24 V) | M (24 V) | LDB 14 |
| 32 | +24 V    | M (5 V)  | + 15 V   | LDB 15 | 32 | +24 V    | M (5 V)  |          | LDB 15 |

Table 4-2 Interface Assignment of the Backplane Connector 2

#### 4.4 Interface Assignment of the Front Connector

| Pin No. | Assignment                      |
|---------|---------------------------------|
| 1       | Housing/ground/M <sub>ext</sub> |
| 2       | RxD                             |
| 3       | VPG + 5 V DC                    |
| 4       | + 24 V from the bus             |
| 5       | Ground/M <sub>int</sub>         |
| 6       | TxD                             |
| 7       | TxD                             |
| 8       | Housing/ground/M <sub>ext</sub> |
| 9       | RxD                             |
| 10      | 24 V ground                     |
| 11      | 20 mA/transmitter               |
| 12      | Ground/M <sub>int</sub>         |
| 13      | 20 mA/receiver                  |
| 14      | VPG + 5 V                       |
| 15      | Ground/M <sub>int</sub>         |

Table 4-3 Interface Assignment of the Front Connector

## Index

### A

- Absolute addressing, 2-10
- Acknowledge monitoring, 1-10
- ADB (See Address bus)
- Address, 1-3, 1-4
  - bits, 1-2
  - bus (ADB), 1-3
  - error, 2-3, 4-1
  - unit, 1-3, 1-4
- Address error (ADF) LED, 2-3
- Addressing
  - absolute, 2-10
  - error (ADF), 2-6, 4-1
  - page frame, 2-10
- "ADF" (See Addressing error)
- "ADF" LED (See Address error LED)
- Altitude, operating, 1-6
- Ambient temperature, 1-6
- Analog input/output modules, 2-10
- Analog inputs, 2-10
- Analog outputs, 2-10
- Application, 1-1
- Arithmetic, floating-point, 1-1
- Arithmetic/logic unit, 1-3, 1-4
- Arithmetic operations, 1-4
- Automatic cold restart, 3-2
- Automatic warm restart, 2-4, 3-2

### B

- Backplane connectors, interface
  - assignment, 4-8
- Backplane housing, 1-2
- Back-up battery, 2-9, 3-2
- Back-up current, 1-6
- BASP (See Disable digital outputs)
- "BASP" LED (See Disable output command LED)
- "BASP" signal, 2-3, 2-4, 2-8, 2-9
- Battery, back-up, 2-9, 3-2
- Battery failure error (BAU), 2-6, 4-4
- BAU error (See Battery failure error)
- Bit operations, 1-4
- Bit slice unit (BSU), 1-3
- Block diagram, 1-2, 1-3
- Blocks
  - data, 1-6
  - extended data, 1-6, 3-2
  - extended function, 1-6
  - function, 1-6
  - organization, 2-4, 2-7
  - program, 2-4
  - sequence, 2-4
- BSU (See Bit slice unit)

### Bus

- address, 1-3
- data, 1-3
- drivers, 1-3
- enable, 1-3
- global, 1-2, 1-3
- interrupts, 1-3
- local, 1-2, 1-3, 4-1
- local address, 1-3
- local data, 1-3

- Bus-lock logic, 1-4

### C

- Central controller housing, 1-8, 2-1, 2-2, 4-3
- Central register for error addresses, 4-1
- Clock
  - real-time, 1-1
  - update, 1-3
- Code, machine, 1-7
- Cold restart, 2-3, 2-5, 2-6, 2-7, 2-8, 3-1, 3-2
  - automatic, 3-2
- Communication
  - backplane housing, 1-2
  - communications processors, 1-1
  - local bus, 1-2
- Communications processors (CPs), 1-1, 1-4, 2-10
- Communication unit, 1-3, 1-4
- Connection
  - operator panel, 1-2
  - programmer, 1-2
- Controller unit, 1-2
- Control RAM, 2-8, 3-1, 3-2
- Control switches, 2-2 to 2-8
- Coordinator 923C, 1-2, 1-3, 2-6, 3-1, 3-2
- Counters, 1-6
- Current, back-up, 1-6
- Current consumption, 1-6
- Cycle time error (ZYK) LED, 2-3, 2-6, 2-8
- Cyclic operation, 2-4, 2-6, 3-1, 3-2
- Cyclic program processing, 1-1, 2-4, 2-5, 2-6, 2-8, 3-1
  - interrupt, 2-4, 2-5, 2-6, 2-8

### D

- Data
  - bits, 1-2
  - block (DB), 1-6
  - block DB1, 1-6, 2-4, 2-6, 2-7, 2-8, 3-2
  - bus (DB), 1-3
  - flow, 1-3
  - user, 1-4
  - width, 1-2



DB (See Data block and Data bus)  
DB1 (See Data block DB1)  
Degree of protection, 1-6  
Description, technical, 1-1 to 1-8  
Design, 1-1  
Digital input/output modules, 1-6, 1-8  
Digital inputs, 2-8, 2-10  
Digital outputs, 2-4, 2-8, 2-10  
    disable (BASP), 2-3, 2-4, 2-8  
    enable, 2-4  
Dimensions, 1-6, 1-7  
Disable digital outputs (BASP), 2-3, 2-4, 2-8  
Disable output command ("BASP") LED, 2-3  
DXs (See Extended data blocks)

## E

Electronics, 1-2  
Enable digital outputs, 2-4  
End process control programmer  
    function, 2-6  
Environmental rating, 1-6  
EPROM  
    submodules, 1-2, 1-4, 1-5, 1-6  
    user, 1-4  
Error addresses, central register for, 4-1  
Error address format, 4-1  
Errors, 2-4, 4-1  
    addressing (ADF), 2-3, 2-6, 4-1  
    battery failure (BAU), 2-6, 4-8  
    cycle time (ZYK), 2-3, 2-6, 2-8  
    hardware, 2-6  
    I/O not operable (PEU), 2-6, 4-8  
    parity (PARE), 2-7, 4-4  
    power failure (NAU), 2-4, 2-6, 4-8  
    reaction to, 2-4  
    substitution (SUF), 2-4  
    system, 2-4, 2-5, 2-7  
    time-out (QVZ), 2-3, 2-4, 2-6, 2-7  
    user program, 3-1, 3-2  
Extended data blocks (DXs), 1-6, 2-6, 3-2  
Extended function blocks (FXs), 1-6

## F

FBs (See Function blocks)  
Flags, 1-6, 2-4  
Floating-point arithmetic, 1-1  
Force programmer function, 2-4  
Front connector, 1-2, 1-4, 3-2, 4-1, 4-9  
    interface assignment, 4-9  
Front panel, 1-2, 2-1, 2-2, 2-3, 2-4, 2-5  
Functional capabilities, 1-8  
Function blocks (FBs), 1-6  
FXs (See Extended function blocks)

## G

Global bus, 1-2, 1-3

## H

Hard STOP LED, 2-3, 2-7  
Hard "STOP" mode, 2-7  
Hardware, 1-1, 1-3, 1-8  
    errors, 2-6  
    interrupts, 1-1  
    structure, 1-3

## I

"INIT FAULT" LED, 2-3, 2-8, 2-9  
Input modules  
    analog, 2-10  
    digital, 1-8, 2-10  
Input/output modules (I/Os), 1-4, 1-8  
Inputs  
    analog, 2-10  
    digital, 2-10  
    interrupt, 1-8  
Installation, 1-8, 2-1 to 2-10  
Integrated RAM, 1-1  
Intelligent input/output modules (IPs), 1-4  
Interface assignment, backplane connectors,  
    4-8  
Interface assignment, front connector, 4-9  
Interrupt, 1-1, 1-3, 1-8, 2-4, 2-7, 2-8  
    bus, 1-3  
    cyclic program processing, 2-4, 2-5, 2-6,  
        2-8  
    hardware, 1-1  
    inputs, 1-8  
    overall reset, 2-4  
    parity, 4-1  
    priorities, 1-1  
    process, 2-4, 2-7, 2-8  
    signals, 4-3  
    time, 2-4, 2-7  
Interrupt driven program processing, 1-1  
Interrupt stack (ISTACK), 2-7, 2-9  
    overflow (STUEU), 2-7  
I/O not operable error (PEU error), 2-6, 4-8  
I/Os (See Input/output modules)  
ISTACK (See Interrupt stack)

## J

Jumpers, layout, 4-2 to 4-7

## K

Kiloword, 1-1  
Kword (See Kiloword)

## L

LAB (See Local address bus)  
LDB (See Local data bus)  
LEDs, 2-2 to 2-10  
    "5 V DC Power Supply o.k.", 2-9  
    "15 V DC/24 V DC Power Supply o.k.", 2-9  
    address error ("ADF"), 2-3, 2-6, 2-8

- cycle time error ("ZYK"), 2-3, 2-6, 2-8
  - disable output command ("BASP"), 2-3
  - error, 2-3
  - hard STOP, 2-3, 2-7
  - "INIT FAULT", 2-3, 2-8, 2-9
  - operating status, 2-5
  - "RUN", 2-3, 2-4, 2-5, 2-9, 3-1, 3-2
  - signal, 2-7
  - smooth STOP, 2-3, 2-4, 2-5, 2-6
  - "STOP", 2-3, 2-4, 2-5, 2-6, 2-9
  - "SYS FAULT", 2-3, 2-5, 2-7, 2-9
  - time-out ("QVZ"), 4-1
  - Local address bus (LAB), 1-3
  - Local bus, 1-2, 1-3, 4-1
    - communication, 1-2
  - Local data bus (LDB), 1-3
  - Locking bar, 2-1, 2-2
  - Locking screw, 2-2, 2-3
- M**
- Machine code, 1-7
  - MAI (See Memory and interface)
  - Maintenance, 4-1 to 4-9
  - MC5
    - operations, 1-3
    - processor, 1-7
  - Memory
    - and interface (MAI), 1-3
    - area, 1-4
    - assignment, 1-1, 1-5
    - capacity, 1-1
    - configuration, 1-5, 1-6
    - EPROM, 1-4
    - extension, 1-6, 2-9
    - pages, 1-4, 1-5
    - RAM, 1-4
    - space, 1-4, 1-5
    - submodules, 1-2, 1-4, 1-5, 1-6, 2-9, 3-2
    - unit, 1-3, 1-4
  - Memory module (355), 1-1, 1-2, 1-4, 1-5, 2-9, 4-1
    - parity interrupt, 4-1
  - Microprocessor (80186), 1-2, 1-4
  - Microprogram control, 1-2
  - Mode
    - S5-150S, 1-8
    - S5-155U, 1-8
    - smooth "STOP," 1-1
  - Modules
    - installing, 2-1 to 2-2
    - removing, 2-1, 2-2
  - Monitoring
    - acknowledge, 1-10
    - parity, 1-1
    - scan time, 2-4, 2-8
  - MUART (See Multifunction universal asynchronous receiver transmitter)
  - Multifunction universal asynchronous receiver transmitter (MUART), 1-4
  - Multiplexing, time-division, 1-3
  - Multiprocessing, 1-1, 1-3, 2-8, 3-1, 3-2
  - Multiprocessor operating, 3-2
- N**
- NAU error (See Power failure error)
  - Ninth track of the control RAM, 2-8
- O**
- OB1 (See Organization block 1)
  - Operating altitude, 1-6
  - Operating status LEDs, 2-5
  - Operating system, 1-4, 1-8, 3-1
  - Operation, 1-8, 2-3, 2-7, 2-9, 2-10, 3-1, 3-2
    - cyclic, 2-4, 2-6, 3-1, 3-2
    - peripheral modules, 2-10
    - single processor, 2-6, 2-7
  - Operations
    - arithmetic, 1-4
    - bit, 1-4
    - MC5, 1-3
    - STEP 5, 1-1, 1-8, 2-3, 2-7
    - STOP (STP, STS, STW), 2-6
    - word, 1-4
  - Operator
    - control, 2-2
    - information, 2-1 to 2-10
  - Operator panel, connection, 1-2
  - Organization blocks, reaction to errors, 2-7, 2-8
  - Organization block 1 (OB1), 2-4
  - OUTP ADDR programmer function, 4-1
  - Output modules
    - analog, 1-6
    - digital, 1-6
  - Outputs
    - analog, 2-10
    - digital, 1-6, 2-4, 2-8
  - Overall reset function, 2-4
    - interrupt, 2-4
- P**
- 757 programmer multiplexer (PG-MUX 757), 1-4
  - Page frame addressing, 2-10
  - PARE (See Parity error)
  - P area, 2-10
  - Parity
    - error (PARE), 2-7, 4-8
    - interrupt, 4-1
    - monitoring, 1-1
  - PC INFO programmer function, 4-1
  - PC STOP programmer function, 2-6
  - Peripheral modules, 2-7, 2-10
    - operation of, 2-10

Peripherals page, 1-4, 1-5  
Peripheral units, 1-4  
PEU error (See I/O not operable error)  
PG-MUX 757 (See 757 programmer multiplexer)  
Power failure error (NAU), 2-4, 2-6  
Printed circuit boards, jumper layout, 4-2 to 4-7  
Process image, 1-6, 2-4, 2-7, 2-8, 3-1, 3-2  
Process interrupts, 2-4, 2-8  
Program  
    system, 2-6, 2-7, 2-9, 3-1  
    user, 1-1, 1-4, 1-6, 3-1, 3-2  
Program blocks (PBs), 2-4  
Programmer  
    connection, 1-2  
    front connector, 3-2  
Programmer functions, 2-4  
    End process control, 2-6  
    FORCE, 2-4  
    OUTP ADDR, 4-1  
    PC INFO, 4-1  
    PC STOP, 2-6  
Programmer interface, 1-2, 1-4, 1-7, 2-3, 3-1, 3-2  
Programmer serial interface, 1-7  
Programming language, STEP 5, 1-1, 1-4, 1-7  
Program processing  
    cyclic, 1-1, 2-4, 2-5, 2-6, 2-8, 3-1  
    interrupt driven, 1-1  
    time driven, 1-1  
    user, 3-1, 3-2  
Protection, degree of, 1-6

## Q

Q area, 1-6  
QVZ error (See Time out error)  
QVZ LED (See time out LED)  
QVZ signal (See Time out signal)

## R

RAM  
    control, 2-8  
    integrated, 1-1  
    memory, 1-4  
    submodules, 1-2, 1-5  
    user, 1-4, 1-5  
Real-time clock, 1-1  
Register, central for error addresses, 4-1  
Relative humidity, 1-6  
Release lever, 2-1, 2-2, 2-3  
Removing modules, 2-1, 2-2  
"RESET" switch, 2-3, 2-4, 2-5, 2-9, 3-1, 3-2  
Restart function, 3-1, 3-2  
    automatic cold, 3-2  
    automatic warm, 2-4, 3-2  
    cold, 3-2

    warm, 3-2  
Restart types, 3-1  
Restart procedure, 2-8, 2-9, 3-1  
    completed, 2-8  
    error, 2-8  
"RUN" LED, 3-1, 3-2  
"RUN/STOP" switch, 3-1, 3-2

## S

S5-135U programmable controller, 2-10  
S5-150U programmable controller, 2-10  
S5-150S controller mode, 1-8  
S5-150U controller mode, 3-2, 4-2  
S5-155U controller mode, 1-8  
SBs (See Sequence blocks)  
Scan time, 2-8  
    monitoring, 2-4, 2-8  
Sequence blocks (SBs), 2-4  
Signal LEDs, 2-7  
Single processor mode, 1-3  
Single processor operation, 2-6, 2-7  
Slot, 2-1, 2-9  
Smooth STOP LED, 2-3, 2-4, 2-5, 2-6  
Smooth "STOP" mode, 1-1  
STEP 5 operations, 1-1, 2-3, 2-7  
STEP 5 programming language, 1-1  
"STOP" LED, 2-3, 2-4, 2-5, 2-6, 2-9  
"STOP" mode, 1-1, 2-4, 2-5, 2-6, 2-7, 2-8, 3-1  
    smooth, 1-1  
STOP operation, 2-6  
STUEU (See Interrupt stack overflow)  
Substitution error (SUF), 2-4  
SUF (See Substitution error)  
Supply voltage, 2-8  
Switches  
    control, 2-2 to 2-8  
    "RESET," 2-3, 2-4, 2-5, 2-9, 3-1, 3-2  
    "RUN/STOP," 3-1, 3-2  
    "SYS FAULT" LED, 2-3, 2-5, 2-7, 2-9  
System  
    data, 1-2, 1-4  
    error, 2-4, 2-5, 2-7  
    operating, 1-2, 1-4, 1-8, 3-1  
    page, 1-4, 1-5  
    program, 3-1  
    structure, 1-1, 1-2

## T

Technical description, 1-1 to 1-8  
Technical specifications, 1-6, 1-7  
Temperature, ambient, 1-6  
Tests, 2-9  
Time-division multiplexing, 1-3  
Time driven program processing, 1-1  
Time interrupts, 2-4, 2-7  
Time-out  
    error (QVZ), 2-3, 2-4, 2-6, 2-7

("QVZ") LED, 4-1  
signal (QVZ), 4-1  
Timers, 1-3, 1-4, 1-6, 2-4  
update, 1-3, 2-4  
Time-slice procedure, 1-3  
Track nine of control RAM, 2-8  
Transmission speed, programmer serial  
interface, 1-2, 1-7

**U**

User data, 1-4  
User EPROM, 1-4  
User memory, 1-1, 1-2, 1-5, 1-6  
capacity, 1-1  
space, 1-5  
User program, 1-1, 1-4, 1-6

error, 3-1, 3-2  
parts, 2-8  
processing, 1-1, 3-1  
RAM, 1-4, 1-5

**V**

Voltage, supply, 2-8

**W**

Warm restart, 3-2  
Weight, 1-7  
Word operations, 1-4

**Z**

"ZYK" LED (See Cycle time error LED)

# SIEMENS

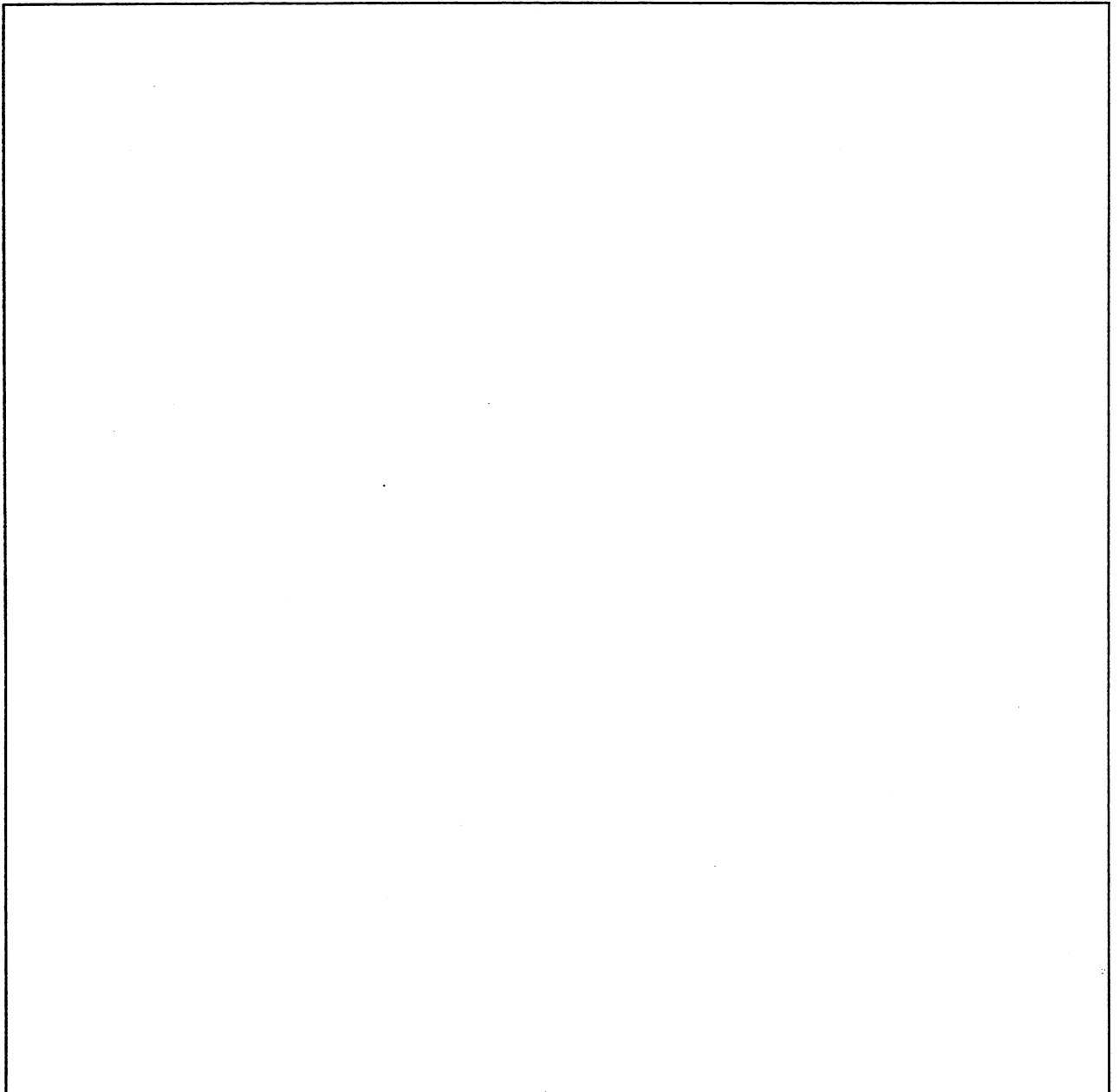
## SIMATIC S5

355 Memory Module

6ES5 355-3UA11

Instructions



C79000-B8576-C382-06





Siemens has developed this document for its licensees and customers. The information contained herein is the property of Siemens and may not be copied, used, or disclosed to others without prior written approval from Siemens. Users are cautioned that the material contained herein is subject to change by Siemens at any time and without prior notice.

Siemens shall not be responsible for any damages, including consequential damages, caused by reliance on material presented, including but not limited to typographical, electronic, arithmetic, or listing errors.

|   |  |
|---|--|
|  |  <b>WARNING</b>   |
|   | <p><b>Hazardous voltage.</b></p> <p><b>Can cause death, severe personal injury, or substantial property damage.</b></p> <p>Restrict use to qualified personnel.<br/>See safety instructions.</p> |

Only qualified personnel should install or maintain this equipment after becoming thoroughly familiar with all warnings, safety notices, and maintenance procedures contained in this manual. The successful and safe operation of this equipment is dependent upon proper handling, installation, operation, and maintenance.

The following are definitions of the terms “qualified person,” “danger,” “warning,” and “caution,” as applicable for this document.

**Qualified Person**

One who is familiar with the installation, construction, and operation of this equipment and the hazards involved. In addition, the person should have the following qualifications:

- Be trained and authorized to use and tag circuits and equipment in accordance with established safety practices
- Be trained in the proper care and use of protective equipment in accordance with established safety practices
- Be trained in rendering first aid

**DANGER**

Indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.

**WARNING**

Indicates that loss of life, 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.

STEP 5® and SIMATIC® are registered trademarks of Siemens AG.



## Preface

This book provides the hardware description of the 6ES5 355-3UA11 memory module and explains its installation and operation.

This book is intended for engineers, programmers, and maintenance personnel who have a general knowledge of programmable controller concepts.

If you have any questions about the 355 memory module not answered in this book, please contact your local Siemens representative.





## How to Use This Book

This section discusses information that may be helpful as you use this book.

### Contents of This Book

- **Chapter 1 - Technical Description**

This chapter describes the application and design of the 6ES5 355-3UA11 memory module and includes general information about memory submodules to be inserted in the 355 memory module.

This chapter also lists the technical specifications for the 6ES5 355-3UA11 memory module.

- **Chapter 2 - Operation**

This chapter discusses installation of the 6ES5 355-3UA11 memory module. It also explains its addressing procedure, operation using back-up battery, and data integrity using parity bits.

- **Chapter 3 - System Start-Up**

This chapter explains how to handle system start-up after you plug in or remove the 6ES5 355-3UA11 memory module or any of its submodules.

- **Chapter 4 - Pin and Jumper Assignment**

This chapter discusses the central register for error addresses with reference to parity errors and explains the contents of the instruction register. It also explains the memory submodule ID and shows the jumper setting, jumper layout, and connector pin assignment of the 6ES5 355-3UA11 memory module and its submodules.

- **Chapter 5 - Accessories/Spare Parts**

This chapter lists spare parts for the 6ES5 355-3UA11 memory module.

- **Index**

The index contains an alphabetical list of key terms and subjects covered in this book and their corresponding page numbers.

- **Remarks Form**

The remarks form is provided for your comments and recommendations.

### **Training**

Contact your local Siemens representative for information on training courses to aid you in becoming familiar with this product. Consult the appendix at the end of the publication no. C79000-B8576-C452 for a list of Siemens offices worldwide.

### **Reference Materials**

It is recommended that you have the following books that support the S5-155U system:

- *Catalog ST 54.1: S5-135U, S5-155U and S5-155H Programmable Controllers*

(Order No. E86010-K4654-A111-A6-7600)<sup>1</sup>

---

<sup>1</sup> Order this book from your local Siemens representative.

Programmer Manuals\*:

- *PG 635 Programmer*  
(Order No. 6ES5 835-0SC21)\*
- *PG 675 Programmer (S5-DOS)*  
(Order No. 6ES5 875-0SC21)\*
- *PG 685 Programmer*  
(Order No. 6ES5 885-0SC21)\*
- *PG 730 Programmer*  
(Order No. 6ES5 884-0FC21)\*
- *PG 750 Programmer*  
(Order No. 6ES5 886-0SC21)\*
- *Programming Package for Personal Computers*  
(Order No. 6ES5 896-0SC21)\*
- *S5-135U (CPU 928)*  
(Order No. 6ES5 998-1UL22)\*
- *S5-135U (S and R Processor)*  
(Order No. 6ES5 998-0UL21)\*
- *U Periphery*  
(Order No. 6ES5 998-0PC22)\*
- *You will find an introduction to programming with STEP 5, as well as an explanation of how to work with the S5-155U programmable controller and its I/O modules in the following book:*

*Automating with the SIMATIC S5-155U*  
by Hans Berger  
Siemens AG, ISBN 3-8009-1562-6

---

\* Order from your local Siemens representative.

**Conventions**

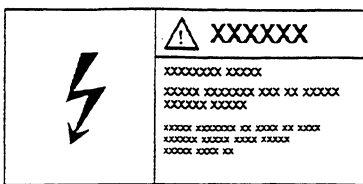
The following conventions are used in this book and are listed for your reference:

**Convention**

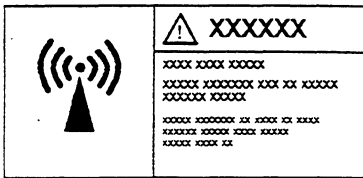
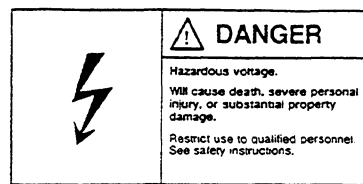
**Definition**

**Example**

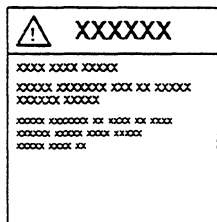
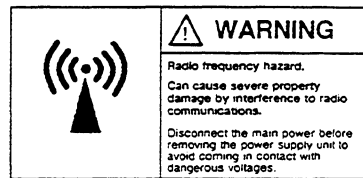
A box that indicates a type of hazard, describes its implications, and tells you how to avoid the hazard, is safety notation. Some safety notation includes a graphic symbol representing an electrical or radio frequency hazard. All safety notation has one of the following levels of caution:



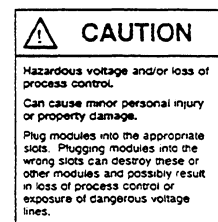
- A danger indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.



- A warning indicates that loss of life, severe personal injury, or substantial property damage can result if proper precautions are not taken.



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



## Contents

|   |         |
|---|---------|
| <b>Preface</b> .....  | 0-1     |
| <b>How to Use This Book</b> .....                           | 0-3     |
| <b>Chapter 1 - Technical Description</b> .....              | 1-1     |
| 1.1 Application .....                                       | 1-1     |
| 1.2 Design .....  | 1-2     |
| 1.3 Technical Data .....                                    | 1-2     |
| <b>Chapter 2 - Operation</b> .....                          | 2-1     |
| 2.1 Addressing .....  | 2-1     |
| 2.2 Operation Using Back-Up Battery .....                   | 2-2     |
| 2.3 Data Integrity Using Parity Bits .....                  | 2-3     |
| <b>Chapter 3 - System Start-Up</b> .....                    | 3-1     |
| <b>Chapter 4 - Pin and Jumper Assignment</b> .....          | 4-1     |
| 4.1 Central Register for Error Addresses .....              | 4-1     |
| 4.2 Jumper Setting of the Memory Module .....               | 4-2     |
| 4.3 Jumper Layout of the Memory Module .....                | 4-2     |
| 4.4 Connector Pin Assignment of the 355 Memory Module ..... | 4-3     |
| <b>Chapter 5 - Accessories/Spare Parts</b> .....            | 5-1     |
| <b>Index</b> .....  | Index-1 |
| <b>Page Overview</b> .....                                  | Index-3 |

**Figures**

4-1 Jumper Layout of the 355 Memory Module ..... 4-2

**Tables**

4-1 Jumper Assignment of the 355 Memory Module ..... 4-2  
4-2 Pin Settings of the 355 Memory Module Backplane  
Connector 1 ..... 4-3  
4-3 Pin Assignment of the 355 Memory Module Backplane  
Connector 2 ..... 4-3  
4-4 Connector Pin Assignment for Submodules 1, 2, and 3 ..... 4-4  
5-1 355 Memory Module Spare Parts/Accessories ..... 5-1

**NOTE**

These instructions do not cover all details or variations in equipment or provide for every circumstance that can arise with installation, operation, or maintenance. If you want further information or if particular problems arise that are not covered sufficiently for your purposes, contact your local Siemens sales office.

The contents of this instruction manual shall not become part of or modify any prior or existing agreement, commitment, or relationship. The sales contract contains the entire obligation of Siemens. The warranty contained in the contract between the parties is the sole warranty of Siemens. Any statements contained herein do not create new warranties or modify the existing warranty.

# Chapter 1

## Technical Description

This chapter describes the application and design of the 6ES5 355-3UA11 memory module (355 memory module).

### 1.1 Application

The 355 memory module and its accompanying RAM and EPROM submodules are used in the S5-155U programmable controller as memory extension for user programs of CPU 946/947 and in the S5-155H for the user programs of the CPU 946R/947R.

You can operate this memory module in the designated slots in the central controller housing of the S5-155U and S5-155H.

You can plug a maximum of three of the following long memory submodules into one 355 memory module:

- RAM submodules -
  - 6ES5 377-0AB21, 16 Kwords (32 Kbytes)
  - 6ES5 377-0AB31, 32 Kwords (64 Kbytes)
  - 6ES5 377-0AB41, 64 Kwords (128 Kbytes)
  
- EPROM submodules -
  - 6ES5 373-0AA41, 16 Kwords (32 Kbytes)
  - 6ES5 373-0AA61, 32 Kwords (64 Kbytes)
  - 6ES5 373-0AA81, 64 Kwords (128 Kbytes)

The RAM submodules extend the available user RAM area in the CPU. The system program software of CPU 946/947 and CPU 946R/947R sets the address areas of the submodules automatically during a cold restart.

The submodules are set only in **word operation**. When programming (blowing) the EPROM submodules at the PG in the S5-DOS package "EEPROM/EPROM" select the operating mode "WORD/FIELD".

The 355 memory module has an integrated memory area for parity monitoring (1 parity bit per byte) for maximum configuration with three 64 Kword<sup>1</sup> submodules.

## 1.2 Design

The electronics of the 355 memory module are on a printed circuit board with a 64-pin connector to link the 355 memory module to the S5 local bus. The front panel of the module is the width of 1 1/3 standard plug-in station and has three receptacles where you can plug in the RAM or EPROM submodules.

## 1.3 Technical Data

|  |   |
|--|---|
| Degree of protection                         | IP <sup>1</sup> 00  |
| Permissible ambient temperature              |   |
| - operation                                  | 0° C to +55° C  |
| - storage and transport                      | - 40° C to +70° C   |
| Relative humidity                            | Maximum 95% at 25° C, noncondensing   |
| Operating altitude                           | Up to 3500 m (11,483.5 ft.) <sup>2</sup> above sea level  |
| Capacity                                     | 3 receptacles for submodules with 16 Kwords, 32 Kwords, or 64 Kwords, configured with EPROM or static RAM memory;<br>3 parity memory areas, each with 128 Kbits |
| Data transmission line                       | 2 x 8 bits bidirectional; S5 local bus, word operation only   |
| Addresses                                    | 20 address lines A0 <sup>3</sup> to A19   |
| Addressing area                              |   |
| - for submodules                             | 10000H to 6FFFFH and<br>80000H to DFFFFH  |
| - decoding RAM                               | E8100H to E817FH  |
| - instruction register,<br>control addresses | E8010H to E8011H<br>E8012H to E8013H  |
| - central error register<br>(in CPU 947)     | E8000H, E8004H  |
| Addressing of the submodules                 | Any given block of 16 Kwords in the address area<br>from 10000H to DFFFFH   |

<sup>1</sup> IP stands for an environmental rating. The first number (0 to 6) after IP is the dry particle rating and the second number (0 to 8) is the moisture rating.

<sup>2</sup> Where dimensions are indicated in meters and feet, the conversion factor used is 3.281 (1 m = 3.281 ft.) with feet rounded off to the nearest tenth of a foot.

<sup>3</sup> A0 is the least significant address.



|   |   |
|---|---|
| Supply voltage  | + 5 V, $\pm$ 5 %  |
| Battery voltage $U_{\text{Batt}}$   | 2.7 V to 4.75 V <sup>1</sup>  |
| Interface level   | TTL   |
| Current consumption   |   |
| - memory module without submodules<br>operating current ( + 5 V)<br>(typically)             | 800 mA; maximum 1200 mA   |
| back-up current ( + 3 V)<br>(typically)   | 10 $\mu$ A; maximum 300 $\mu$ A   |
| - memory module with three RAM<br>submodules<br>operating current ( + 5 V)<br>(typically)   | 900 mA; maximum 1400 mA   |
| back-up current ( + 3 V)<br>(typically)   | 20 $\mu$ A; maximum 900 $\mu$ A   |
| - memory module with three EPROM<br>submodules<br>operating current ( + 5 V)<br>(typically) | 1000 mA; maximum 1800 mA  |
| back-up current<br>(typically)  | 10 $\mu$ A; maximum 300 $\mu$ A   |
| Access time   |   |
| - read  | $\leq$ 320 nsec. after trailing edge of LMEMR                                 |
| - write   | $\leq$ 320 nsec. after trailing edge of LMEMW                                 |
| - address search  | $\geq$ 80 nsec.   |
| - LMEMR or LMEMW is deactivated   | $\geq$ 70 nsec. after RDY   |
| - access time to modules  | $\leq$ 250 nsec.  |
| Scan time $t_{\text{cyc}}$<br>(read or write)   | $\geq$ 560 nsec.  |
| Dimensions (w x h x d)  | 20.32 mm (0.79 in.) x 233.4 mm (9.10 in.) x 160 mm<br>(6.24 in.) <sup>2</sup> |
| Weight  | Approximately 400 g (0.88 lb.) <sup>3</sup>                                   |

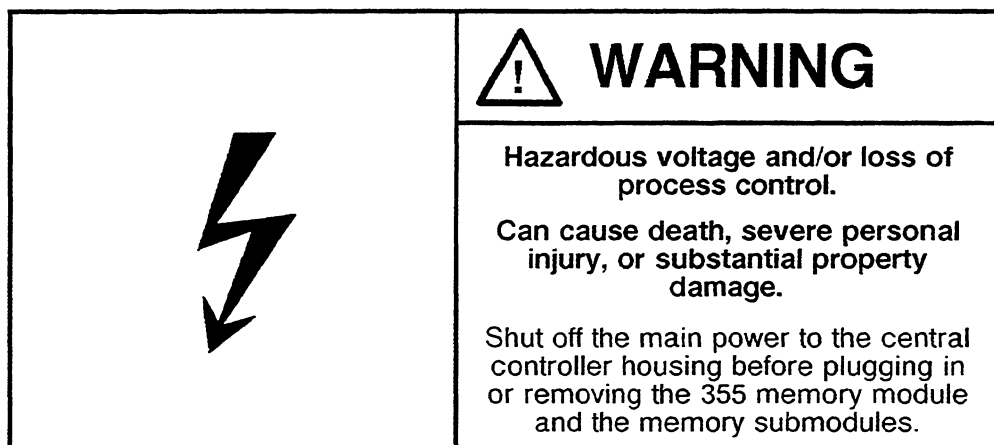
<sup>1</sup> The parity memory area and RAM submodules are backed up by the battery in the central controller housing.

<sup>2</sup> Where dimensions are indicated in millimeters and inches, the conversion factor used is 0.039 (1 mm = 0.039 in.) with inches rounded off to the nearest hundredth of an inch.

<sup>3</sup> Where weights are indicated in kilograms and pounds, the conversion factor used is 2.2 (1 kg = 2.2 lbs.) with pounds rounded off to the nearest tenth of a pound.

## Chapter 2 Operation

The S5-155U housing has specific slots for inserting the 355 memory module.



You can plug two 355 memory modules into each CPU 946/947. See the *S5-155U Central Controller Housing Hardware and Installation Guide* for details. Empty module slots can be provided with a cover. (See Section 5, Accessories/Spare Parts).

### 2.1 Addressing

Each time CPU 946/947 starts up, its system program handles addressing automatically. It first reads out the IDs of all memory submodules that are plugged in (coded in the hardware of their connectors) and the instruction register of the 355 memory module(s). Then it writes the start addresses of all submodules into the decoding RAM of each 355 memory module. The system program arranges the RAM submodules with consecutive addresses in ascending order, the EPROM submodules with consecutive addresses in descending order.

The start addresses can be found in the chapter Memory Assignment in the CPU 946/947 or CPU 946R/947R Instructions.

## 2.2 Operation Using Back-Up Battery

The 355 memory module is buffered using the back-up battery of the S5-155U central controller (see the *S5-155U Central Controller Instructions*).

Back-up refers to maintenance of data stored in the static RAMs when the supply voltage of the programmable controller is switched off.

A backup is implemented for the parity storage of the 355 memory module and the inserted RAMs. A monitor circuit in the 355 memory module handles the switchover from the 5-V power supply to the back-up battery.

A monitor circuit in the power supply unit detects low battery voltage and reports it to the power supply using the LED on the front panel. Depending on the jumper setting on the power supply, this message can also be output at the alarm relay. After mains ON the power supply reports this fault (BAU) to the CPU. The CPU then goes into the "STOP" mode (see the *S5-155U Central Controller Instructions and the CPU 946/947, eliminating faults, BAU*).

You will lose your data if you remove RAM submodules that have not been backed up or if you remove the 355 memory module during operation using the back-up battery.

---

## 2.3 Data Integrity Using Parity Bits

Two parity bits (one bit for each data byte) protect all data words of the memory submodules. These parity bits are stored in a static RAM that can be backed up. The system program generates the parity bits automatically during an overall reset of CPU 946/947 and also during a cold restart of CPU 946/947 for submodules that have just been plugged in. This applies also for RAM and EPROM submodules.

The CPU detects any parity errors when it reads the memory. The newly updated parity bits of data read are compared to the stored parity bits. If the parity bits in question are not equal, the error is reported to CPU over a special signal line on the local bus (PARE, see Table 4-3). At the same time, the address where the first parity error occurred is stored in a central error register in the CPU.

You can disable the parity error signal by inserting jumper X-7 (see Table 4-1) or by using the system program via the instruction register of the 355 memory module. After you plug in a memory module or submodule, you must enable the system program to regenerate the parity bits (e.g., with a cold restart).

### NOTE

**Do not address EPROM submodules with a write request (possible using STEP 5 operations with absolute addresses). Doing so overwrites the parity bits. An attempt to read under such an address results in a parity error.**

## Chapter 3 System Start-Up

You can configure the 355 memory submodules according to your needs. Do not change any jumper settings on the memory submodules.

The operating system detects any changes in the memory caused when you remove or plug in memory submodules or a 355 memory module with submodules. If an error occurs the red LED "INITFAULT" on the CPU 947 flashes.

If there is any change in submodule configuration, CPU 946/947 goes into the "STOP" mode when power is restored. The "STOP" LED flashes slowly or quickly, depending on the change in submodule configuration. The way in which it flashes indicates what type of start-up is possible (i.e., cold restart or overall reset).

After a change in memory configuration, the following restart types are possible:

|   |                            |
|---|----------------------------|
| Removing EPROM submodules<br>or the memory module<br>with EPROM submodules            | → Cold restart required    |
| Removing RAM submodules<br>or the memory module<br>with RAM submodules                | → Overall restart required |
| Plugging in EPROM/RAM submodules<br>or the memory module<br>with EPROM/RAM submodules | → Cold restart required    |
| Removing/plugging in the<br>memory module without<br>submodule configuration          | → Warm restart permissible |

After you complete the overall reset procedure, you can load your user program while the CPU is in the "STOP" mode or during cyclic processing (i.e., "RUN" mode).

## **Chapter 4**

### **Pin and Jumper Assignment**

This chapter discusses briefly the central register for error addresses. The possible jumper assignment, layout, and connector pin assignment of the 355 memory module are shown here.

#### **4.1 Central Register for Error Addresses**

When a parity interrupt occurs, the address where the first error appears is stored in a central register located in the module 947. If the module 947 detects a time-out error when it attempts to access the local bus, the central register for error addresses contains the first QVZ (time-out) address.

When you read out error addresses the parity interrupt that was set in CPU 947 is reset. Reading out does not change (reset) the contents of the central register for error addresses.

### 4.2 Jumper Settings

The jumper settings may not be changed. The only exception is jumper 7; see Chapter 2.3.

| Jumper and Setting |     | Function                              |
|--------------------|-----|---------------------------------------|
| In                 | Out |                                       |
| X7                 |     | Parity interrupt output is suppressed |
|                    | X7  | Parity interrupt output is enabled    |

Table 4-1 Jumper Settings of the 355 Memory Module

### 4.3 Jumper Layout

Figure 4-1 shows the jumper layout of the 355 memory module.

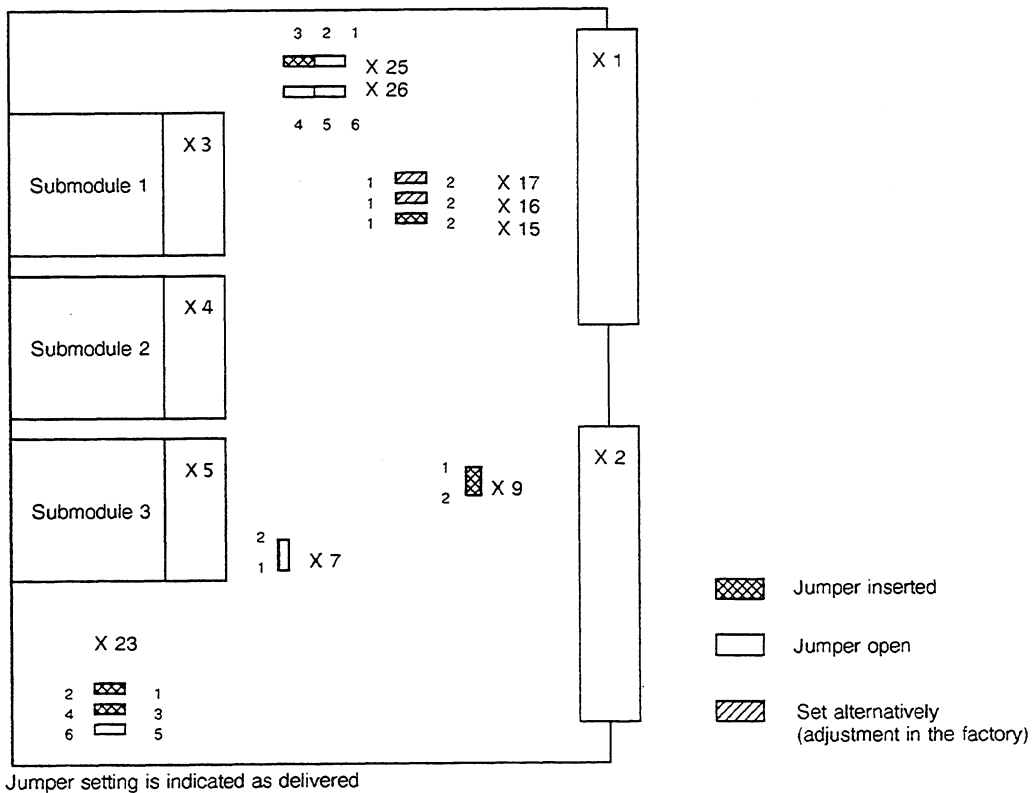


Figure 4-1 Jumper Layout of the 355 Memory Module

#### 4.4 Connector Pin Assignment of the 355 Memory Module

Tables 4-2 and 4-3 show the pin assignment of the 355 memory module backplane connectors 1 and 2.

|    | z     | b | d          | f      |
|----|-------|---|------------|--------|
| 2  | + 5 V | M | M          | LAB 0  |
| 4  | + 5 V |   | UBAT       | LAB 1  |
| 6  | CPKL  |   |            | LAB 2  |
| 8  |       |   |            | LAB 3  |
| 10 |       |   |            | LAB 4  |
| 12 |       |   |            | LAB 5  |
| 14 |       |   |            | LAB 6  |
| 16 |       |   |            | LAB 7  |
| 18 |       |   |            | LAB 8  |
| 20 |       |   |            | LAB 9  |
| 22 |       |   |            | LAB 10 |
| 24 |       |   |            | LAB 11 |
| 26 |       |   |            | LAB 12 |
| 28 |       |   | <u>DSI</u> | LAB 13 |
| 30 |       |   | STK        | LAB 14 |
| 32 |       | M |            | LAB 15 |

Table 4-2 Pin Assignment of the 355 Memory Module Backplane Connector 1

|    | z           | b | d            | f      |
|----|-------------|---|--------------|--------|
| 2  | + 5 V       | M | LAB 16       | LDB 0  |
| 4  |             |   | LAB 17       | LDB 1  |
| 6  |             |   | M            | LDB 2  |
| 8  |             |   | LAB 18       | LDB 3  |
| 10 |             |   | LAB 19       | LDB 4  |
| 12 | M           |   | <u>LMEMR</u> | LDB 5  |
| 14 |             |   | <u>LMEMW</u> | LDB 6  |
| 16 |             |   | <u>LRDY</u>  | LDB 7  |
| 18 | +5 V        | M |              | LDB 8  |
| 20 |             |   |              | LDB 9  |
| 22 |             |   |              | LDB 10 |
| 24 |             | M |              | LDB 11 |
| 26 | <u>PARE</u> |   |              | LDB 12 |
| 28 | +5 V        |   |              | LDB 13 |
| 30 |             |   |              | LDB 14 |
| 32 |             | M |              | LDB 15 |

Key to Tables 4-2 and 4-3:

- LAB 0 to 19 = Local Bus Addresses
- LDB 0 to 15 = Local Bus Data
- DSI = Data Save
- LMEMR = Local Read
- LMEMW = Local Write
- LRDY = Local Ready
- PARE = Parity Error
- CPKL = Clear
- STK = Slot ID
- UBAT = Back-Up Voltage
- + 5 V = Supply Voltage
- M = Chassis Ground (0 V)

Table 4-3 Pin Assignment of the 355 Memory Module Backplane Connector 2



## Pin and Jumper Assignment

---

Table 4-4 shows the connector pin assignment for submodules 1, 2, and 3.

| Pin | C                | B                | A       |
|-----|------------------|------------------|---------|
| 1   | SADB 12          | M                | + 5 V   |
| 2   | SADB 0           | SADB 1           | SADB 2  |
| 3   | SADB 3           | SADB 4           | SADB 5  |
| 4   | SADB 6           | SADB 7           | SADB 8  |
| 5   | SADB 9           | SADB 10          | SADB 11 |
| 6   | SADB 13          | SADB 14          | RD      |
| 7   | PGM/WR           | SDBH 0           | SDBH 1  |
| 8   | SDBH 2           | SDBH 3           | SDBH 4  |
| 9   | SDBH 5           | SDBH 6           | SDBH 7  |
| 10  | SDBL 0           | SDBL 1           | SDBL 2  |
| 11  | SDBL 3           | SDBL 4           | SDBL 5  |
| 12  | SDBL 6           | SDBL 7           | K1      |
| 13  | $\overline{CS1}$ | $\overline{CS3}$ | K2      |
| 14  | $\overline{CS2}$ | CS4              | K3      |
| 15  | UCMOS            | PSW              | K4      |
| 16  | VPP              | M                | K5      |

Table 4-4 Connector Pin Assignment for Submodules 1, 2, and 3

## Chapter 5 Accessories/Spare Parts

Table 5-1 lists accessories/spare parts for the 355 memory module. Order spare parts from your local Siemens representative.

| Name   | Order Number                                       | Spare Parts Group        |
|--|--|--------------------------|
| RAM submodules <ul style="list-style-type: none"> <li>● 16 Kwords</li> <li>● 32 Kwords</li> <li>● 64 Kwords</li> </ul>   | 6ES5 377-0AB21<br>6ES5 377-0AB31<br>6ES5 377-0AB41 | R <sup>1</sup><br>R<br>R |
| EPROM submodules <ul style="list-style-type: none"> <li>● 16 Kwords</li> <li>● 32 Kwords</li> <li>● 64 Kwords</li> </ul> | 6ES5 373-0AA41<br>6ES5 373-0AA61<br>6ES5 373-0AB81 | R<br>R<br>R              |
| Coding plugs<br>(jumpers)  | W79070-G2602-N2                                    | N <sup>2</sup>           |
| Cover for module slot  | C79451-A3079-C251                                  | N                        |

<sup>1</sup> R = can be repaired

<sup>2</sup> N = cannot be repaired

Table 5-1 355 Memory Module Spare Parts/Accessories

## A

Access time  
 address search, 1-3  
 LMEMR OR LMEMW deactivated, 1-3  
 read, 1-3  
 to modules, 1-3  
 write, 1-3

Addresses, 1-2  
 central register for, 4-1  
 control, 1-2  
 EPROM submodules, 2-1  
 lines, 1-2  
 RAM submodules, 2-1

Addressing, 2-1  
 modules, 1-2

Addressing area  
 control addresses, 1-2  
 decoding RAM, 1-2  
 instruction register, 1-2  
 submodules, 1-1, 1-2

Address search, access time, 1-3

Altitude, operating, 1-2

Ambient temperature, 1-2

Application, 1-1

## B

Backup, 2-2

Back-up battery, 2-2  
 operation, 2-2

Back-up current, 1-3

Battery back-up  
 parity memory area, 1-2  
 RAM submodules, 1-2

Battery, back-up, 2-2

Battery voltage, 1-3

Bus, S5 local, 1-2

## C

Capacity, 1-2

Central controller housing, 1-1

Central register for error addresses, 4-1

Coding plugs, 5-1

Cold restart, 3-1

Configuration, submodules, 1-2

Connector pin assignment  
 module, 4-3  
 submodules, 4-4

Control addresses, addressing area, 1-2

Cover for module slot, 2-1, 5-1

## Current

back-up, 1-3  
 operating, 1-3

Current consumption, 1-3

## D

Data  
 integrity, 2-3  
 transmission line, 1-2

Decoding RAM, 1-2, 2-1  
 addressing area, 1-2

Degree of protection, 1-2

Description, technical, 1-1 to 1-2

Design, 1-2

Dimensions, 1-3

## E

Electronics, 1-2

Environmental rating, 1-2

EPROM submodules, 1-1, 1-2, 5-1  
 addresses, 2-1  
 parity bits, 2-3

Error  
 parity, 4-1  
 time-out (QVZ), 4-1

Error addresses, central register for, 4-1

## F

Front panel, 1-2, 2-2

## H

Housing, central controller, 1-1

## I

IDs, submodules, 2-1

Instruction register, 2-1, 2-3  
 addressing area, 1-2

Interface level, 1-3

Interrupt, parity, 4-1, 4-2

## J

Jumper  
 assignment, 4-1  
 layout, 4-2  
 settings, 3-1, 4-2

## L

LMEMR or LMEMW deactivated,  
 access time, 1-3

Local bus, S5, 1-2

## M

Memory, changes in, 3-1

Modules

- access time to, 1-3
- addressing, 1-2
- connector pin assignment, 4-1, 4-3
- plugging in, 3-1
- removing, 3-1

Monitor circuit

- low battery voltage, 2-2
- switchover from power supply to back-up battery, 2-2

Monitoring, parity, 1-1

## O

Operating altitude, 1-2

Operating current, 1-3

Operation, 2-1 to 2-3

- using back-up battery, 2-2

Overall reset, 3-1

## P

PARE (See Parity error signal line)

Parity bits, 2-3,

- EPROM submodules, 2-3
- generation, 2-3
- overwriting, 2-3
- RAM submodules, 2-3
- regeneration, 2-3

Parity errors, 2-3, 4-1

Parity error signal line (PARE), 2-3

- disable, 2-3

Parity interrupt, 4-1

- enabled, 4-2
- reset, 4-1
- set, 4-1
- suppressed, 4-2

Parity memory areas, 1-2

- battery back-up, 1-3

Parity monitoring, 1-1

Parity storage, 2-2

Plugging in

- modules, 3-1
- submodules, 3-1

Programs, user, 1-1

Protection, degree of, 1-2

## Q

QVZ (See Time-out error)

## R

RAM, decoding, 2-1

RAM submodules, 1-1, 1-2, 5-1

- addresses, 2-1

- battery back-up, 1-3

- parity bits, 2-3

Read, access time, 1-3

Relative humidity, 1-2

Removing

- modules, 3-1
- submodules, 3-1

Reset, overall, 3-1

Restart

- cold, 3-1
- warm, 3-1

“RUN“ mode, 3-1

## S

S5-155U programmable controller slots,

- 1-1, 2-1

S5 local bus, 1-2

Scan time, 1-3

Slots of the S5-155U programmable controller,

- 1-1, 2-1

Spare parts, 5-1

Start addresses, submodules, 2-1

“STOP“ mode, 3-1

Submodules

- addressing areas, 1-1, 1-2
- connector pin assignment, 4-4
- EPROM, 1-1, 1-2, 5-1
- IDs, 2-1,
- maximum configuration, 1-1
- memory, 1-1
- plugging in, 3-1
- RAM, 1-1, 1-2, 5-1
- removing, 3-1
- start addresses, 2-1

Supply voltage, 1-3

System start-up, 3-1

## T

Technical description, 1-1

Technical data, 1-2 to 1-3

Temperature, ambient, 1-2

Time-out error (QVZ), 4-1

Transmission line, data, 1-2

## U

User programs, 1-1

User RAM area, extension, 1-1

## V

Voltage

- battery, 1-3
- supply, 1-3

## W

Warm restart, 3-1

Weight, 1-3

Write, access time, 1-3

## SIMATIC S5 923 C Coordinator

6ES5 923-3UC11

Instructions

C79000-B8576-C349-06

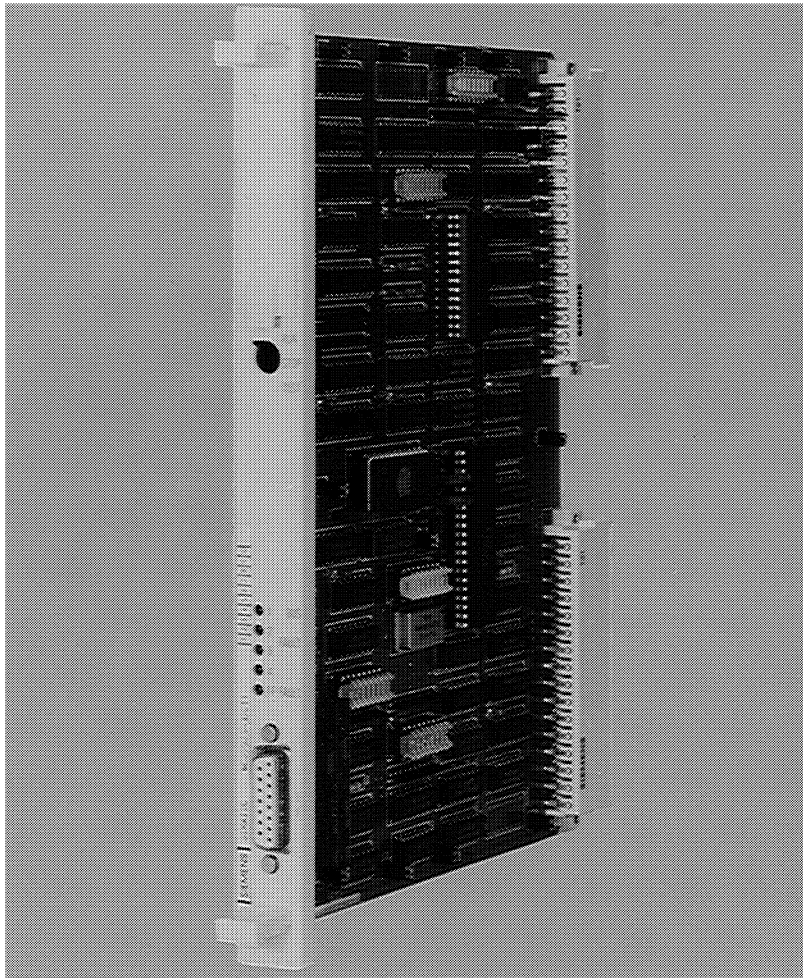


Fig. 1 923 C Coordinator

| Contents                                  | Page     | Page   | Page      |
|---|----------|--|-----------|
| <b>1 Technical Description</b>            | <b>3</b> | <b>3 Operation</b>                                 | <b>9</b>  |
| 1.1 Application                           | 3        | 3.1 Displays and Operator Controls                 | 9         |
| 1.2 Design                                | 3        | 3.2 Operating Modes                                | 9         |
| 1.3 Mode of Operation                     | 4        | 3.3 Setting the Coordination                       | 10        |
| 1.3.1 Bus Arbitration                     | 4        | 3.4 Setting the PG Multiplexer                     | 11        |
| 1.3.2 Monitoring Bus Occupation Time      | 5        | 3.5 Addressing the Mailbox                         | 13        |
| 1.3.3 Mailbox                             | 5        | 3.6 Jumpers for Disabling the Coordination Signals | 14        |
| 1.3.4 PG Multiplexer                      | 6        | 3.7 Error Register                                 | 14        |
| 1.4 Technical Data                        | 7        | 3.8 Location of Switches & Jumpers                 | 15        |
| <b>2 Installation</b>                     | <b>8</b> | <b>4 Maintenance</b>                               | <b>16</b> |
| 2.1 Removing and Inserting the Module     | 8        | 4.1 Connector Pin Assignment                       | 16        |
| 2.2 Slots in the Programmable Controllers | 8        | 4.2 Spare Parts List                               | 17        |
|   |          | <b>5 Page Overview</b>                             | <b>18</b> |

## 1 Technical Description

### 1.1 Application

The 923 C coordinator (COR) is installed in the S5-135 U and S5-155U programmable controllers (except -3KA12 and -3KB12) and in the expansion units EG 185U and EG 186U. It is primarily intended to perform three independent tasks:

- o Bus arbitration

The coordination of multiprocessing, i.e. the simultaneous use of two to four CPUs (S processor/GPU 921, R processor/GPU 922, M processor/CPU 920, CPU 928 or CPU 946/947).

- o Mailbox

For the exchange of data between CPUs.

- o Programmer multiplexer (PG MUX)

Central access to the serial PG interfaces of maximum eight modules in the programmable controllers (PLC) or expansion units (EG) with the corresponding programmers (PG) via the serial interface of the COR 923 C.

With this function the COR 923 C can be connected to the factory bus SINEC H1 via the communications processor CP 535.

The PG software S5-DOS is required to operate the PG MUX.

For more information on the multiprocessor functions please refer to the Instructions for Multiprocessing in the S5-135U and S5-155U, ref. no. C79000-B8576-C500 and the User's Guide for Multiprocessor Communication, ref. no. C79000-B8576-C468.

### 1.2 Design

The COR 923 C is a plug-in PCB in double Euroformat.

Two 48-pin blade connectors of the "row 2" type connect the module to the S5 bus in the subrack.

The width of the front panel takes up 1 1/3 standard slots.

In the upper third of the front panel there is a recess with a cover. By removing this cover the DIL (dual in-line) switches for the assignment of parameters to the module are accessible.

A toggle switch with three settings is also mounted on the front panel for other operator control functions.

There are five small LED's which indicate errors or reactions to errors.

The COR 923 C can be connected to the required programmers, diagnostic units, the operator panel or the CPs by means of a 15-pin front connector.

### 1.3 Mode of operation

#### 1.3.1 Bus arbitration

- Bus enable signals

The COR 923 C enables each of the two to four CPUs in the S5-135 U or S5-155U to use the bus cyclically. Each CPU can only use the common S5 bus during this time.

The bus enables are assigned in a time-division multiplex operation. The number of CPUs can be adjusted with DIL switches on the COR 923 C. The enable time for accessing the S5 bus is fixed at 2  $\mu$ s for all the CPUs. The bus enable time can be extended with a bus lock (carried out automatically by the CPU).

The order of the bus assignments begins with CPU 1 after the reset signal is cleared by the power supply and, according to the number of CPUs set, consecutively enables:

CPU 1, CPU 2, CPU 3, CPU 4, CPU 1, CPU 2 etc.

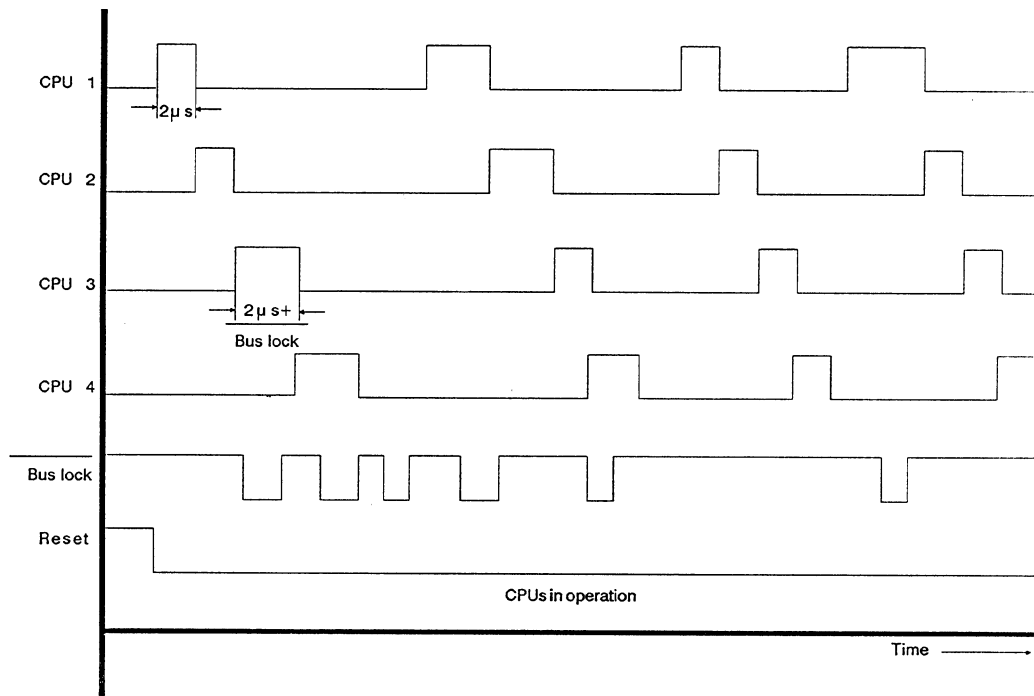


Fig. 2 Operational sequences of the bus control signals

### 1.3.2 Monitoring bus occupation time

The signal for the bus lock can only be output by the CPU which has already been given a bus enable by the COR. The bus enable time for the CPU is extended by the duration of the bus lock signal (see Fig. 2). The factory setting of the bus lock signal monitoring is 2 ms. If the signal stays active longer than this, the COR 923 C outputs a signal which stops all the CPUs.

The processor, whose bus lock signal exceeded the maximum time, is indicated in a register which can be read by the S5-135 U under the address FEFFH, and the corresponding "BUS FAULT" LED on the front panel of the COR 923 C lights up. The register and the LED are cleared again when the signal which caused the stop goes inactive.

### 1.3.3 Mailbox

A mailbox on the COR 923 C assumes among other things the function of the interprocessor communication (IPC) flags. The IPC flags make possible the cyclic exchange of data between the CPUs. The mailbox also contains four page frames for the exchange of data blocks between the CPUs.

How these two functions are programmed can be found in the Programming instructions for the CPUs.

The mailbox comprises a RAM buffered centrally via the programmable controllers.

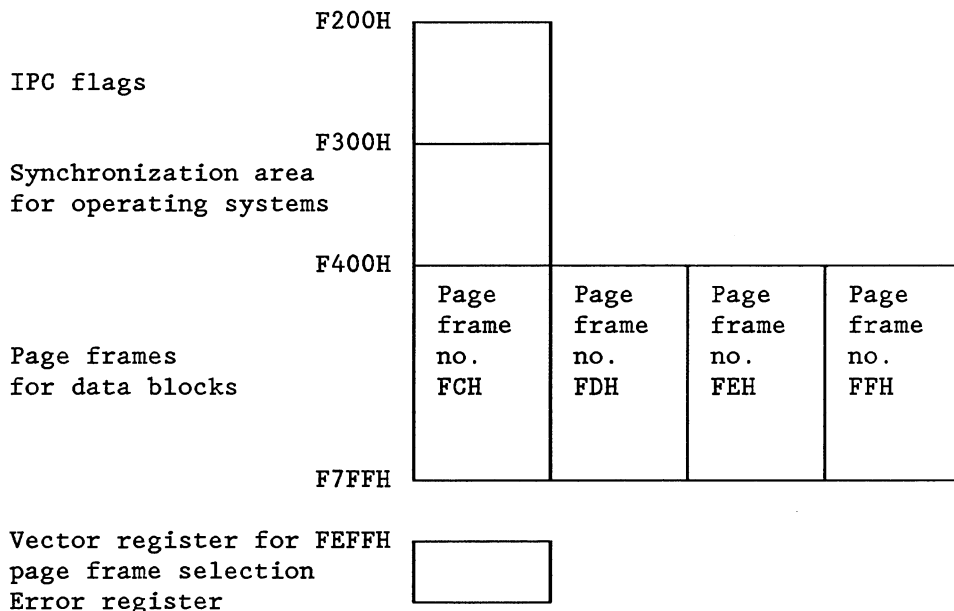


Fig. 3 Memory areas of the mailbox on the S5 bus



- Addressing procedure for the page frame (vector register)

The vector register is used to create subaddresses of several memories in a common address area. The register is an 8-bit register, which can be written into under the address FEFFH. It cannot be read back.

There are four page frames each with  $1 \times 2^{10}$  bytes. Each page frame has an ID number allocated to it. The numbers are FFH, FEH, FDH, FCH. These numbers have a fixed setting on the COR 923 C and cannot be changed.

Every access to the page frame area must be immediately preceded by the loading of one of these four numbers in the vector register, in order to select the page frame.

These numbers must not be used on other modules (CP, IP) in the same PLC, otherwise double addressing will result.

After the supply voltage is switched on, the vector register is erased. The vector register then contains the number 0H.

The transfer of data to and from this memory is implemented by special CPU functions. These functions can be found in the relevant programming instructions. The S processor CPU 921 does not possess these functions.

#### 1.3.4 PG multiplexer

The TTY interface on the front of the COR 923 C can be switched to eight different serial interfaces in the PLCs with the aid of an intelligent multiplexer.

These multiplex interfaces are TTL level and are wired to the other modules via backplane connector 2 and the bus PCB.

- Procedure for the selection of serial interfaces

All the modules in the PLCs served by the multiplexer have individual numbers allocated to them. These numbers must be between 1 and 31 (decimal). The lowest of these numbers, the base address, is set in binary using the DIL switch S2. The maximum of eight numbers are allocated to the S5-135 U slots 11, 19, 27, 35, 43, 51, 59 and 67 (the lowest number to slot 11).

All eight numbers (or slots) are assigned to switch S3, the lowest to S3.1, the highest to switch S3.8.

If slots are not occupied or if modules are to be operated using their own front connectors, the numbers allocated to the particular slots must be masked out with switch S3.

The front connector of the PLC interface of a module being served by the multiplexer must remain unused.

**1.4 Technical data**

|   |   |
|---|---|
| Degree of protection                                  | IP 00   |
| Operational temperature                               | 0 to 55°C   |
| Transport and storage temperature                     | -40 to 70°C   |
| Relative humidity                                     | Max. 95% at 25°C<br>No condensation                       |
| Operating altitude                                    | Max. 3500 m above sea level                               |
| Power supply voltage                                  | 5 V $\pm$ 5%<br>24 V +25%/-15%                            |
| Current consumption at 5 V                            | Typ. 1.1 A  |
| Current consumption at 24 V                           | 60 mA   |
| Minimum back-up battery voltage for CMOS RAM          | 2.7 V   |
| Back-up current                                       | Typ. 2 $\mu$ A  |
| Acknowledgement time for access to mailbox via S5 bus | Typ. 320 ns   |
| Signalling rate of the serial interface               | 9600 bit/s  |
| Transmission cable                                    | Screened 4-wire-line<br>PG connecting cable<br>6ES5 731-1 |
| Transmission distance                                 | Max. 1 km at 9600 bit/s                                   |
| Weight  | Approx. 0.3 kg  |
| Dimensions (W x H x D)                                | 20.32 mm x 233.4 mm x 160 mm                              |

## **2 Installation**

### **2.1 Removing and inserting the module**

The module is removed from the front of the central controller by gently rocking it up and down using the handles. Modules can only be removed or inserted when the central controller is switched off (power supply off).

### **2.2 Slots in the programmable controllers**

Multiprocessor operation and PG MUX:

- in the S5-135U, slot 3
- in the S5-155U, slot 3

PG MUX only:

- in the EG 185U, slot 11
- in the EG 186U, slot 19.

### 3 Operation

#### 3.1 Displays and operator controls

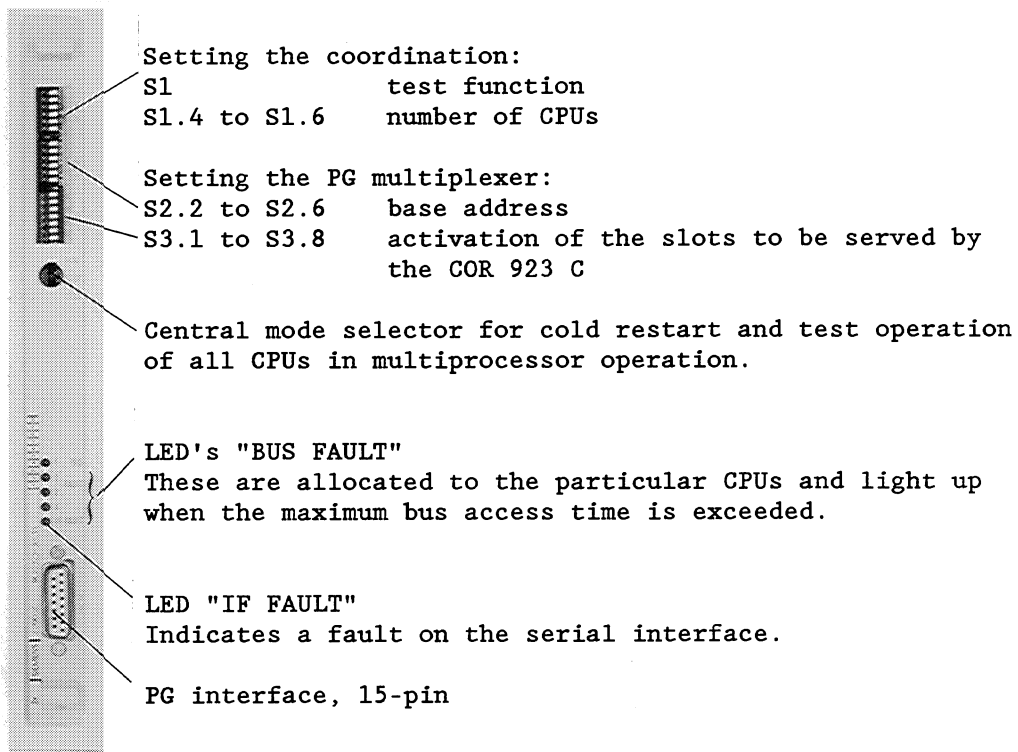


Fig. 4 Front panel of the COR 923 C

#### 3.2 Operating modes

o Stop status

If the mode selector is at "STOP" after the supply voltage has been switched on or if another stop request is present, the CPUs remain in the stop status.

o Cold restart

A cold restart is carried out when the position of the mode selector is changed from "STOP" to "RUN" providing the supply voltage has been switched on, and the CPUs have been individually reset.

### 3.4 Setting the PG multiplexer

#### o Base address

The DIL switch S2 is used to set a base address between 1 and 31 (necessary for connection to the factory bus, since several PGs may have access). The modules selected by the multiplexer can be addressed under this and the following seven addresses. The base address is the sum of the binary values activated by setting the switch position "on".

As delivered:

|      | off | on |                            |
|------|-----|----|----------------------------|
| S2.1 | x   |    | —                          |
| 2    | x   |    | Value 16                   |
| 3    | x   |    | Value 8                    |
| 4    | x   |    | Value 4                    |
| 5    | x   |    | Value 2                    |
| 6    |     | x  | Value 1 (base address = 1) |

#### o Activation of addresses

The numbers or slots which are to be operated from the COR 923 C must be activated with switch S3.

As delivered:

|      | off | on |                  | Slot no.<br>(in S5-135 U) |
|------|-----|----|------------------|---------------------------|
| S3.1 | x   |    | Base address + 0 | 11/CPU 1                  |
| 2    | x   |    | Base address + 1 | 19/CPU 2                  |
| 3    | x   |    | Base address + 2 | 27/CPU 3                  |
| 4    | x   |    | Base address + 3 | 35/CPU 4                  |
| 5    | x   |    | Base address + 4 | 43                        |
| 6    | x   |    | Base address + 5 | 51                        |
| 7    | x   |    | Base address + 6 | 59                        |
| 8    | x   |    | Base address + 7 | 67                        |

**Example**

The modules in slots 11, 35, 43 and 67 in the S5-135 U are to be addressed via the COR 923 C using the base address 10.

Setting the base address:

|      | off | on |                   |
|------|-----|----|-------------------|
| S2.1 | x   |    | —                 |
| 2    | x   |    | Value 16          |
| 3    |     | x  | Value 8 → 8       |
| 4    | x   |    | Value 4           |
| 5    |     | x  | Value 2 → + 2     |
| 6    | x   |    | Value 1 —         |
|      |     |    | 10 (Base address) |

Activation of the required slots:

|      | off | on |                  | Slot no.<br>(in S5-135 U) | Slots to be<br>operated | Addr. |
|------|-----|----|------------------|---------------------------|-------------------------|-------|
| S3.1 |     | x  | Base address + 0 | 11                        | 11                      | 10    |
| 2    | x   |    | Base address + 1 | 19                        |                         |       |
| 3    | x   |    | Base address + 2 | 27                        |                         |       |
| 4    |     | x  | Base address + 3 | 35                        | 35                      | 13    |
| 5    |     | x  | Base address + 4 | 43                        | 43                      | 14    |
| 6    | x   |    | Base address + 5 | 51                        |                         |       |
| 7    | x   |    | Base address + 6 | 59                        |                         |       |
| 8    |     | x  | Base address + 7 | 67                        | 67                      | 17    |

### 3.5 Addressing the mailbox

The IPC flag area extends from F200H to F2FFH; this corresponds to 256 bytes. It can be set in 32 byte steps. Without CPs all 256 bytes are enabled for operation in the S5-135 U and the S5-155U are available for IPC flag functions.

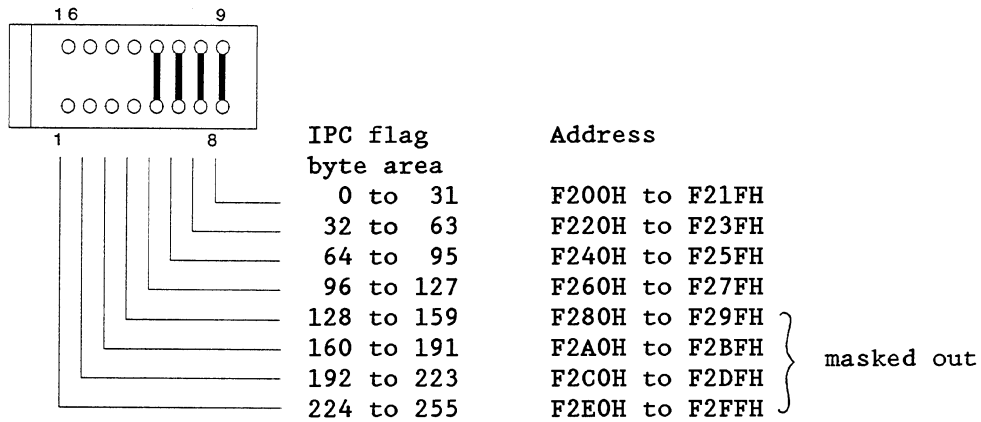
By removing jumpers on slot 60, one or several of the 32 byte areas can be masked out.

If IPC flag bytes are used on a CP, the corresponding areas must be masked out on the COR.

#### Example

The four IPC flag areas with the highest addresses are to be masked out:

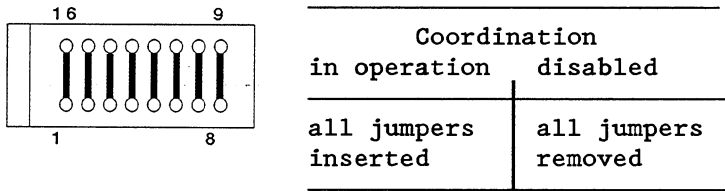
Slot 60



### 3.6 Jumpers for disabling the coordination signals

The COR 923C can also be used as a PG MUX in the expansion units EG 185U and EG 186U. In this mode of operation the coordination signals must be disabled.

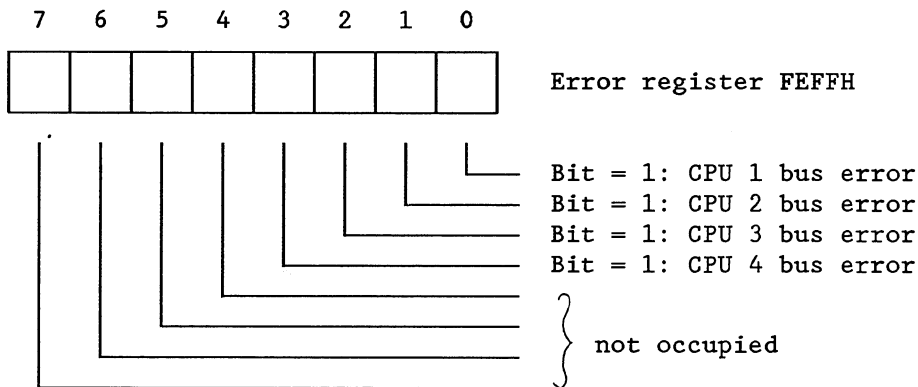
Slot 61



When delivered all jumpers are inserted.

### 3.7 Error register

The error register is an 8-bit register which can be read on the CPU side under the address FEFFH. An entry is made in the register by the bus monitoring if a bus error occurs. One bit in the error register is allocated to each CPU, and this is set to 1 if an error occurs. The register is erased whenever the halt signal becomes inactive.

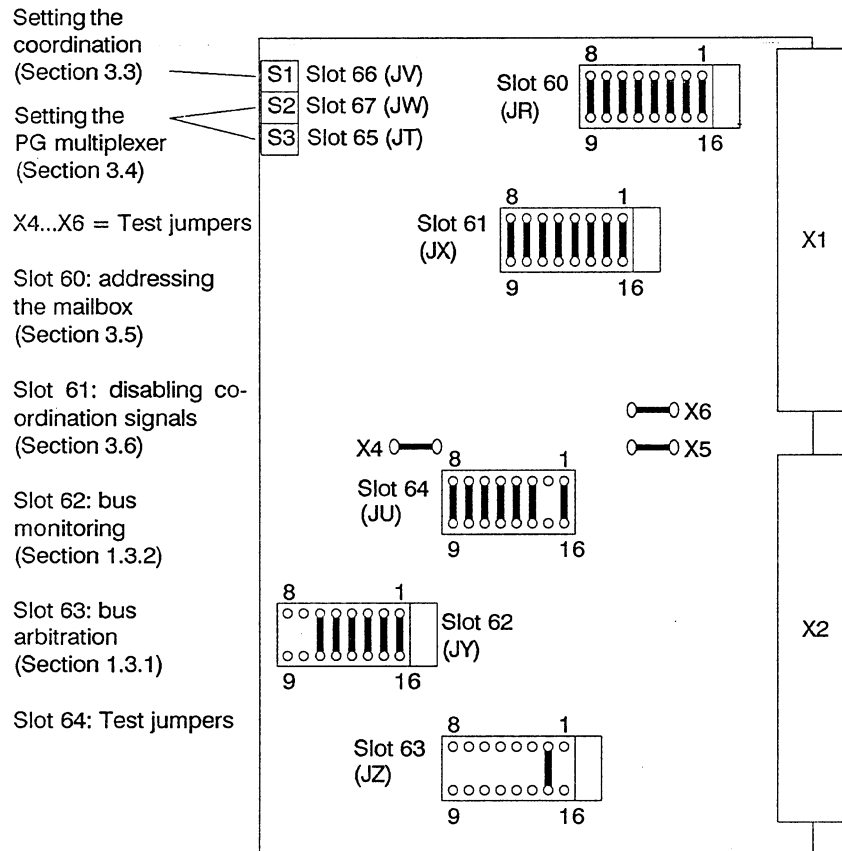


The error register can be read by all CPUs so that central functions can be initiated.



### 3.8 Location of the switches and jumpers

The jumper settings shown correspond to those at the time of delivery.



The jumpers X4 to X6 and the slots 62 to 64 must not be changed.

## 4 Maintenance

### 4.1 Connector pin assignment

Backplane connector 1

|    | d          | b      | z           |
|----|------------|--------|-------------|
| 2  |            | M 5 V  | +5 V        |
| 4  | U BAT      |        |             |
| 6  | ADB 12     | ADB 0  | <u>CPKL</u> |
| 8  | ADB 13     | ADB 1  | <u>MEMR</u> |
| 10 | ADB 14     | ADB 2  | <u>MEMW</u> |
| 12 | ADB 15     | ADB 3  | RDY         |
| 14 | BUSEN 1    | ADB 4  | DB 0        |
| 16 | BUSEN 2    | ADB 5  | DB 1        |
| 18 | BUSEN 3    | ADB 6  | DB 2        |
| 20 | BUSEN 4    | ADB 7  | DB 3        |
| 22 |            | ADB 8  | DB 4        |
| 24 |            | ADB 9  | DB 5        |
| 26 |            | ADB 10 | DB 6        |
| 28 | <u>DSI</u> | ADB 11 | DB 7        |
| 30 |            | BASP   |             |
| 32 |            | M 5 V  | <u>HALT</u> |

Backplane connector 2

|    | d            | b            | z          |
|----|--------------|--------------|------------|
| 2  |              | M 5 V        | +5 V       |
| 4  |              |              |            |
| 6  | <u>RXD 8</u> |              |            |
| 8  | <u>TXD 8</u> |              |            |
| 10 | <u>RXD 7</u> |              |            |
| 12 | <u>TXD 7</u> | <u>RXD 6</u> |            |
| 14 | <u>RXD 5</u> | <u>TXD 6</u> | <u>NAU</u> |
| 16 | <u>TXD 5</u> | <u>RXD 4</u> |            |
| 18 | <u>RXD 3</u> | <u>TXD 4</u> |            |
| 20 | <u>TXD 3</u> | <u>STEU</u>  |            |
| 22 | <u>RXD 1</u> | <u>STOPA</u> |            |
| 24 | <u>TXD 1</u> | <u>RXD 2</u> |            |
| 26 | <u>TEST</u>  | <u>TXD 2</u> |            |
| 28 |              | PERO         |            |
| 30 |              |              | M 24 V     |
| 32 |              | M 5 V        | +24 V      |

Front connector

| Pin | Designation                            |
|-----|--|
| 1   | Frame (F <sub>ext</sub> )              |
| 2   | Receiver TTY (-)                       |
| 3   | PL <sup>1)</sup>                       |
| 4   | +24 V                                  |
| 5   | PL <sup>1)</sup>                       |
| 6   | Transmitter TTY (+)                    |
| 7   | Transmitter TTY (-)                    |
| 8   | Frame (F <sub>ext</sub> )              |
| 9   | Receiver TTY (+)                       |
| 10  | 24 V Frame (current sources (-) 20 mA) |
| 11  | Current source (+) 20 mA               |
| 12  | PL <sup>1)</sup>                       |
| 13  | Current source (+) 20 mA               |
| 14  | PL <sup>1)</sup>                       |
| 15  | PL <sup>1)</sup>                       |

1) PL = private line

## 4.2 Spare parts list

|             |                   |
|-------------|-------------------|
| Coding plug | C79334-A3011-B12  |
| Front cover | C79451-A3079-C251 |

## SIMATIC S5

### Multiprocessing in the S5 135 U and S5 155 U Programmable Controllers

---

Instructions

Publication No. C79000-B8576-C500-02

---

| Contents                                    | Page      |
|---|-----------|
| <b>1 Introduction</b>                       | <b>2</b>  |
| <b>2 Commissioning</b>                      | <b>4</b>  |
| 2.1 Requirements                            | 4         |
| 2.2 Procedure                               | 4         |
| 2.3 Operating Modes of the Coordinator      | 11        |
| 2.3.1 Normal Operation                      | 11        |
| 2.3.2 Test Operation                        | 11        |
| 2.4 The PG Multiplexer on the Coordinator C | 12        |
| <b>3 Troubleshooting</b>                    | <b>14</b> |





---

Siemens has developed this document for its licensees and customers. The information contained herein is the property of Siemens and may not be copied, used, or disclosed to others without prior written approval from Siemens. Users are cautioned that the material contained herein is subject to change by Siemens at any time and without prior notice.

Siemens shall not be responsible for any damages, including consequential damages, caused by reliance on material presented, including but not limited to typographical, electronic, arithmetic, or listing errors.

---

|   |  |
|---|--|
|  |  <b>WARNING</b>   |
|   | <p><b>Hazardous voltage.</b></p> <p><b>Can cause death, severe personal injury, or substantial property damage.</b></p> <p>Restrict use to qualified personnel.<br/>See safety instructions.</p> |

Only qualified personnel should install or maintain this equipment after becoming thoroughly familiar with all warnings, safety notices, and maintenance procedures contained in this manual. The successful and safe operation of this equipment is dependent upon proper handling, installation, operation, and maintenance.

The following are definitions of the terms "qualified person," "danger," "warning," and "caution," as applicable for this document.

#### **Qualified Person**

One who is familiar with the installation, construction, and operation of this equipment and the hazards involved. In addition, the person should have the following qualifications:

- Be trained and authorized to use and tag circuits and equipment in accordance with established safety practices
- Be trained in the proper care and use of protective equipment in accordance with established safety practices
- Be trained in rendering first aid

#### **DANGER**

Indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.

#### **WARNING**

Indicates that loss of life, 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.

---

STEP 5® and SIMATIC® are registered trademarks of Siemens AG.

---

Copyright © Siemens AG 1989  
First Printing, September 1989  
Printed in the Federal Republic of Germany

---



## 1 Introduction

The S5 135 U and S5 155 U belong to the family of SIMATIC S5 programmable controllers.

The controllers can be used in single processor and multiprocessor operation with up to 4 CPU's.

### The following CPU's are available for the S5 135 U:

- CPU 921 (S processor):** ideal for fast binary signal processing (control tasks);  
programming language is STEP 5
- CPU 922 (R processor):** ideal for fast word processing (closed-loop control, calculation, monitoring, signalling);  
programming language is STEP 5
- CPU 928:** for fast word processing and for very fast binary signal processing;  
programming language is STEP 5
- CPU 920 (M processor):** for programming with higher programming languages (BASIC, C) and Assembler, for arithmetic, sorting and statistics related to S5 system functions (e.g., cold restart, warm restart, communication, process image).

### The following CPU's are available for the S5 155 U:

- CPU 946/947:** for very fast word and binary signal processing, especially double-word and floating-point processing, as well as for extensive programs that have large memory requirements;  
programming language is STEP 5.

You can also use **CPU 922 (R processor)**, **CPU 928** and **CPU 920 (M processor)** in the S5 155 U.

You can plug the CPU's into the programmable controller in any combination as long as the CPU slots are wide enough for the CPU front panel and the S5 bus assignment is appropriate for the CPU. Slot requirements and configuration restrictions are as follows:

- CPU 920, CPU 921 and CPU 922 each require one slot.
- CPU 928 requires two slots.
- CPU 946/947 can require a maximum of five slots (three for CPU 946/947 and one each for a maximum of two 355 memory modules). Only two locations in the S5 155U central controller housing can accommodate this CPU.

In multiprocessor operation, every CPU processes its individual user program, independent of the other CPU's.

The data exchange with I/O modules, CP's, IP's and other CPU's is executed via the common S5 bus. In multiprocessor operation, a coordinator controls CPU access to the S5 bus. The instructions for Coordinator 923C and Coordinator 923A explain how bus allocation works.

Data exchange between CPU's in multiprocessor operation can take place by means of one of the following:

- "Interprocessor Communication (IPC) Flags":

for cyclic exchange of binary data (see the programming guide for the CPU)

- "Special Functions for Multiprocessor Communication":

for program-controlled exchange of entire data blocks. This exchange can be accomplished only with Coordinator 923C and not with CPU 921 (S processor). For more information, see the user instructions for multiprocessor communication in this manual.

The following describes the basic procedure for commissioning the S5 135 U and S5 155 U for multiprocessor operation (section 2).

Section 3 explains several reactions to faults/errors and points to possible causes.

## 2 Commissioning

This section describes the commissioning of the S5 135 U and S5 155 U for multiprocessor operation. The reactions of modules to the individual operating steps are also listed.

### 2.1 Requirements

The following explanation assumes an understanding of operating and programming the individual modules in single processor operation. For further information refer to the instructions in the manuals of the S5 135 U and S5 155 U.

Debugged programs and fully functional CPU's are a further precondition.

### 2.2 Procedure

The S5 135 U and S5 155 U each can accommodate a maximum of four CPU's. The instructions for the S5 135 U and the hardware and installation guide for the S5 155 U indicate the slots where you can plug in CPU's.

A coordinator module is necessary for multiprocessor operation. For the S5 135 U, you can use either Coordinator 923A or Coordinator 923C. For the S5 155 U, you can use Coordinator 923C only.

The coordinator allocates a time slice to each CPU. During this time slice, the CPU can access the S5 bus. The coordinator contains the global memory for data exchange between the CPU's via IPC flags. Coordinator 923C also contains an additional memory with four page frames for the multiprocessor communication function as well as the multiplexer for the serial programmer interface (PG MUX).

#### **Important!**

**In coordinator C and in coordinator A the fixed bus enabling time must *not* be changed!**

#### Note on coordinators:

Plugging a coordinator into the S5 135 U or S5 155 U central controller requires that all CPU's are in multiprocessor operation. Even if you operate the coordinator with only one CPU, multiprocessing conditions apply (e.g., DB1 is necessary, and DX0 is necessary for CPU 946/947 so that it operates in S5 155 U controller mode only).



Commissioning can be divided into 6 steps:

● Step 1:

At the coordinator, set the number of CPU's<sup>1</sup> and enable the IPC flags.

a) Coordinator A (6ES5923-3UA11)

Coding by means of jumpers on the coding socket EP 62:

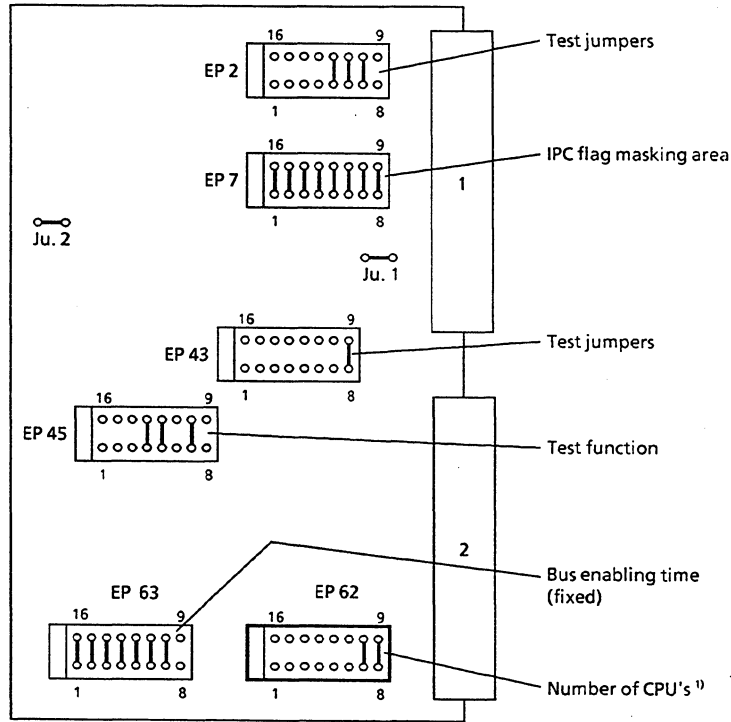


Fig. 1 Position of coding sockets on coordinator A

| Number of CPU's <sup>1</sup> | Jumper(s) on EP 62 |
|------------------------------|--------------------|
| 2                            | 7-10, 8-9          |
| 3                            | 7-10               |
| 4                            | 8-9                |

**IMPORTANT!**

CPU 928 occupies two slots. When you use the S5 135 U in multiprocessor operation, you must always plug in this CPU to the right of the other CPU's. This ensures that CPU's are plugged in from left to right with no intermediate CPU slots empty. When you use the S5 155 U in multiprocessor operation, you must make sure you plug CPUs into the proper slots (see the S5-155U Central Controller Housing Hardware and Installation Guide for slot assignments).

<sup>1</sup> CPU 946/947 counts as one CPU.

b) Coordinator C (6ES5923-3UC11)

Coding carried out by switching on one (only) of the DIL switches S1.4, S1.5 or S1.6 in the front panel recess.

|      | on | off |                                    |
|------|----|-----|------------------------------------|
| S1.1 |    | x   | -                                  |
| 1.2  |    | x   | -                                  |
| 1.3  |    | x   | Test operation (see section 2.3.2) |
| 1.4  | x  |     | Number of CPU's <sup>1</sup> = 2   |
| 1.5  |    | x   | Number of CPU's <sup>1</sup> = 3   |
| 1.6  |    | x   | Number of CPU's <sup>1</sup> = 4   |

(As supplied from the factory, the coordinator is set to 2, i.e. DIL S1.4 is on)

It may also be necessary to address the mailbox on the coordinator.

The 256 IPC flag bytes can be masked in groups of 32 by removing jumpers on the coding socket EP 7 for coordinator A (see Fig. 1 for position) or EP 60 for coordinator C (see Fig. 2).

It is necessary to mask the individual IPC flags if they are used on CP's. In this case, it is essential to mask the (relevant) areas to avoid double addressing (refer to appropriate manuals).

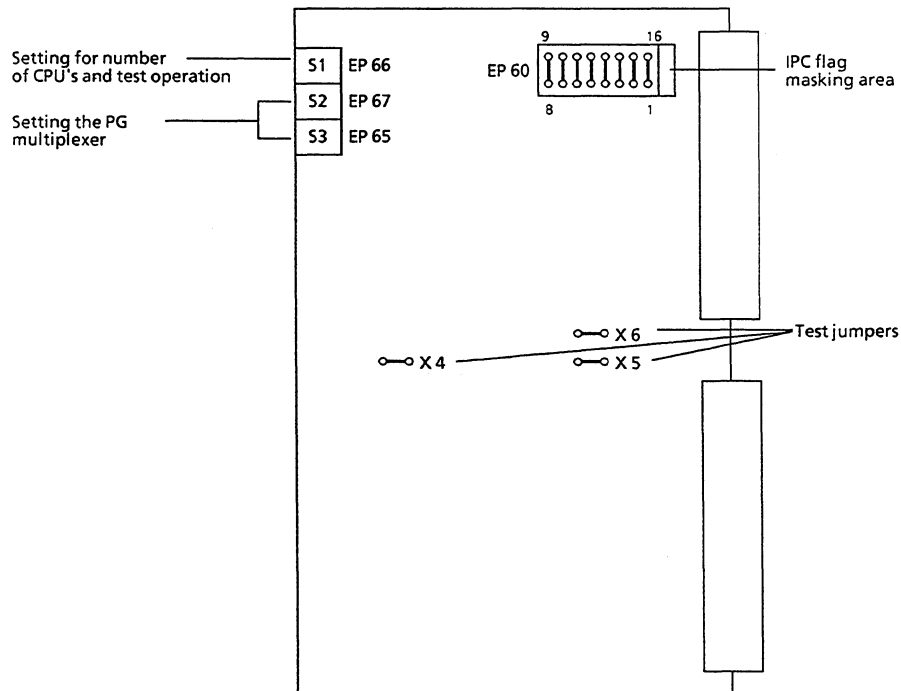
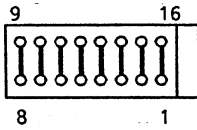


Fig. 2 Position of the coding socket on coordinator C

<sup>1</sup> CPU 946/947 counts as one CPU.



(As supplied from the factory, the coordinator has all IPC flag areas unmasked) (see jumper illustration above)

EP 60 for coordinator C  
EP 7 for coordinator A

| Jumper | IPC flag byte | Address        |
|--------|---------------|----------------|
| 8-9    | 0 to 31       | F200H to F21FH |
| 7-10   | 32 to 63      | F220H to F23FH |
| 6-11   | 64 to 95      | F240H to F25FH |
| 5-12   | 96 to 127     | F260H to F27FH |
| 4-13   | 128 to 159    | F280H to F29FH |
| 3-14   | 160 to 191    | F2A0H to F2BFH |
| 2-15   | 192 to 223    | F2C0H to F2DFH |
| 1-16   | 224 to 255    | F2E0H to F2FFH |

Jumper closed: area unmasked (area acknowledges in the coordinator)

Jumper open: area masked (area acknowledges in the CP)

## ● Step 2:

**With the central controller switched off, plug the CPU's and coordinator into the appropriate slots in the central controller housing. Then turn on the main power.**

When plugging in CPU's in the S5 135 U, fill in the CPU slots from left to right, making sure no intermediate CPU slot is empty. To accomplish this in the S5-135 U, you must plug in CPU 928 to the right of any other CPU's. See the S5-155U, Central Controller Housing Hardware and Installation Guide for slot assignments in the S5 155 U.

Insert all memory submodules (EPROM or RAM) in the CPU's or the 355 memory module (for CPU 946/947), depending on the configuration you use. You must program EPROM submodules on a programmer before inserting them.

Although it is not necessary, it is recommended that you set the mode selector on each CPU and the coordinator to "STOP". Then you can switch on the main power.

## Reaction

After the power supply has been switched on, all CPU's require an "overall reset", i.e. the red STOP LED flashes quickly. Every CPU outputs the BASP signal (provided the test function is not set on the coordinator and the mode selector of the coordinator is not set to "TEST" (see section 2.3.2). This disables digital outputs.

The red BASP LED on the front panel of the CPU indicates this.

● **Step 3:**

**Set mode selector of the coordinator to "STOP" (if you did not do so in Step 2) and carry out an overall reset of all CPU's:**

i.e.: switch the mode selector from "STOP" to "RUN" and back to "STOP" while holding the operating mode switch in the "OVERALL RESET" position.

Repeat this procedure.

**Reaction**

The CPU's with their mode selectors set to "STOP" display a steady red STOP LED. Every CPU continues to output the BASP signal (the BASP LED is on).

● **Step 4:**

**Loading user programs in the inserted RAM submodule**

If you did not plug EPROM submodules containing your user program into the CPU(s) in Step 2, you must load your program into the RAMs.

You can use a combination of CPU's that run with EPROM and with RAM, or with both memory types simultaneously (CPU 946/947 only).

Before a multiprocessing controller can go into cyclic operation, the peripheral allocation must be programmed in each CPU. DB1 must be loaded. For CPU 946/947, you must also assign parameters in DX0 for multiprocessing (S5 155 U controller mode). You can load any additional user program into the CPU's after Step 6 (in cyclic operation).

In the CPU 920 (M processor), the user program can only be loaded if the mode selector is set to "RUN". When switching from "STOP" to "RUN" in order to load the program, the steady light of the STOP LED will stay on in multiprocessor operation (in single processor operation, the steady light changes to a slower flashing light and only changes back to steady light when the program is being loaded).

**Reaction**

No changes from the reactions following step 3.

● **Step 5:**

**For CPU 920, CPU 922, CPU 928 and CPU 946/947, execute a cold re-start. For CPU 921 (S processor), execute a cold restart.**

This means that the mode selector of all CPU's must be switched from "STOP" to "RUN" while the operating mode switch is held in the "RESET" position.

**Reaction**

The red STOP LED on every CPU displays a steady signal. Every CPU also outputs the BASP signal.

● **Step 6:**

**a) Switch mode selector of the coordinator from "STOP" to "RUN"**

**b) If the coordinator is set to test operation (see section 2.3.2) the mode selector can also be set to "TEST".**

**Reaction**

In both cases, the green RUN LED's of all CPU's are steady. All CPU's change over to cyclic operation simultaneously. The BASP signal is not output (the BASP LED is off).

If the coordinator is not set to test operation and if the mode selector is switched from "STOP" to "TEST" there is no reaction.

**Note on LED's**

During the start-up phase (processing of OB 20, OB 21, OB 22) the "STOP" and "RUN" LED's of CPU 920, CPU 922, CPU 928, and CPU 946/947 remain off for a short time. The "RUN" LED only shows a steady light once the central processor changes to cyclic program processing. The "RUN" LED of the CPU 921 (S processor), however, shows a steady light during the start-up phase.

The "STOP" and "RUN" LED's of the CPU 920, CPU 922, CPU 928, CPU 946/947 may therefore remain off during normal coordinator operation (refer to section 2.3.1) while the "RUN" LED of CPU 921 (S processor) is already lit steadily (in actual fact it also waits until every central processor has completed its start-up).

Note on coordinator C

If the coordinator C is used without the PG interface plugged into the front panel, the coordinator C outputs "IF FAULT". In this case, the message can be ignored.

## 2.3 Operating Modes of the Coordinator

### 2.3.1 Normal Operation

The switchover of the individual CPU's to cyclic program execution is synchronized (unless data block DX 0 of the CPU 922, CPU 928, and CPU 946/947 has been programmed differently, see programming instructions for the appropriate CPU), i.e. after every CPU has completed its start-up, all CPU's change to cyclic program execution simultaneously.

If the mode selector of the coordinator is set to "RUN" an error in one CPU (causing it to stop) would stop all other CPU's as well. The STOP LED of the CPU(s) responsible for the stoppage flashes slowly, the STOP LED's of the other CPU's show a steady light.

In addition to the possible display of error LED's on the CPU responsible for the stoppage all CPU's output the BASP signal.

### 2.3.2 Test Operation

By closing the jumper 3-14 on the coding socket EP 45 on coordinator A or by switching on the DIL switch S1.3 on coordinator C test operation can be enabled.

If the mode selector of the coordinator is switched from "STOP" to "TEST", the CPU's can be operated individually. The switchover to cyclic program execution is therefore not synchronized. The output of the BASP signal is suppressed on all CPU's (even if an error occurs!).

If an error occurs on a CPU which is set to "RUN" during test operation, *only this* CPU will stop. The error is indicated by the slow flashing of the STOP LED on the CPU. The error on this CPU does not, therefore, affect the other CPU's.

#### **Warning!**

**Since no CPU can output the BASP signal in the event of an error during test operation, it is essential to deactivate the test mode before putting into operation to prevent operator errors!**

If the test function is deactivated a switchover from "STOP" to "TEST" does not produce any reaction in the CPU's.

## 2.4 The PG Multiplexer on the Coordinator C

The PG multiplexer on the coordinator C allows central access to the serial PG interfaces of up to 8 modules in the PC via the serial PG interface on the front panel of the coordinator.

Operating the multiplexer requires the use of the PG software S5-DOS. However, CPU 920 (M processor) can at present *not* be accessed using this method.

The instructions for the S5 135 U and the hardware and installation guide for the S5 155 U indicate the slots that can be reached via the PG multiplexer and coordinator.

An identification number, which must lie between 1 and 31 (decimal), is allocated to each of these modules. The lowest number, the base address, is always allocated to slot 11. The identification numbers of the other modules are allocated as shown in the following tables.

| S5 135 U: |                    | S5 155 U: |                    |
|-----------|--------------------|-----------|--------------------|
| Slot      | Identification no. | Slot      | Identification no. |
| 11        | Base address       | 11/27     | Base address       |
| 19        | Base address + 1   | 51/67     | Base address + 1   |
| 27        | + 2                | 91        | + 2                |
| 35        | + 3                | 99        | + 3                |
| 43        | + 4                | 107       | + 4                |
| 51        | + 5                | 115       | + 5                |
| 59        | + 6                | 123       | + 6                |
| 67        | + 7                | 131       | + 7                |

The base address is set at the DIL switch S2, in the recess of the front panel of the coordinator C. The base address is the sum of binary values set by the switches in the "ON" position.

|      | off | on |          |
|------|-----|----|----------|
| S2.1 | x   | -  |          |
| 2.2  | x   |    | Value 16 |
| 2.3  | x   |    | 8        |
| 2.4  | x   |    | 4        |
| 2.5  | x   |    | 2        |
| 2.6  |     | x  | 1        |

(As supplied from factory) (In this case, base address = 1)

### IMPORTANT!

**When setting the base address, make sure that no identification number is larger than 31.**

Switch S3, which is also installed in the recess of the front panel of the coordinator C, is used to enable the slots which are to be operated from the PG multiplexer.



**Example for S5 135 U:**

|      | off | on |                          |
|------|-----|----|--------------------------|
| S3.1 | x   |    | slot 11 (first CPU slot) |
| 3.2  | x   |    | 19 (second CPU slot)     |
| 3.3  | x   |    | 27 (third CPU slot)      |
| 3.4  | x   |    | 35 (fourth CPU slot)     |
| 3.5  | x   |    | 43                       |
| 3.6  | x   |    | 51                       |
| 3.7  | x   |    | 59                       |
| 3.8  | x   |    | 67                       |

(As supplied from factory: all slots disabled)

If slots are not occupied or if modules are to be operated via their own front connector these slots must be disabled (= masked). It is especially necessary to mask the slot when using a CPU 920 (M processor).

The front connector of the interface must not be inserted when the module is operated by the multiplexer.

**Note:** while the "ON" position of DIL switch S1 is to the left, it is to the right for switches S2 and S3.

S5-DOS must be used with the programmer. The "bus select" package establishes the requested link between PG and module (for further details, refer to operating instructions of the appropriate programmer).

**Example of a setting on the coordinator C (S5 135U):**

|      | off | on |          |
|------|-----|----|----------|
| S2.1 | x   |    | -        |
| 2.2  | x   |    | value 16 |
| 2.3  |     | x  | 8 → 8    |
| 2.4  |     | x  | 4 → 4    |
| 2.5  | x   |    | 2        |
| 2.6  |     | x  | 1 → 1    |

Base address 13

|      | off | on | Slots accessible from PG | Their address |
|------|-----|----|--------------------------|---------------|
| S3.1 |     | x  | 11                       | 13            |
| 3.2  | x   |    |                          |               |
| 3.3  | x   |    |                          |               |
| 3.4  | x   |    |                          |               |
| 3.5  |     | x  | 43                       | 17            |
| 3.6  |     | x  | 51                       | 18            |
| 3.7  | x   |    |                          |               |
| 3.8  |     | x  | 67                       | 20            |

### 3 Troubleshooting

Section 2 describes the correct commissioning of an S5 135 U or S5 155 U for multiprocessing operation. This section is provided as an aid to troubleshooting.

#### A ... after step 2

##### **Fault/error reaction:**

The "RUN" and "STOP" LED's of some CPU's remain off. The remaining CPU's request "overall reset".

All CPU's output the BASP signal,

##### **Check:**

*Setting of the number of CPU's<sup>1</sup> on the coordinator.*

*Are there gaps between the CPU's in the S5 135 U?*

*Are the CPU's in the appropriate slots in the S5 155 U?*

#### B ... after step 5

##### **Fault/error reaction 1:**

The STOP LED of one CPU flashes slowly. In the control bit display (can be read using PG function "ISTACK") of this CPU a DB 1 error is marked (in addition to usual information). No interrupt stack (ISTACK) page is output.

##### **Check:**

*Has data block DB 1 of this CPU been programmed?*

##### **Fault/error reaction 2:**

After executing a cold restart:

- the CPU 920 (M processor) immediately outputs the QVZ (time-out) signal (as well as BASP) - the red STOP LED flashes slowly,
- the other CPU's (other types) remain stopped (STOP LED on steady) until the last processor goes through a cold restart (with reset).

Then: - **CPU 922 (R processor) or CPU 928:**  
STOP LED flashes slowly

- **CPU 921 (S processor):**  
STOP LED on steady

In the control bit displays of the S5 135 U (can be read using PG) ANL-ABB (termination during start-up) and DB 1 error are marked in addition to the usual entries.

The order in which cold restarts are carried out is immaterial.

If the mode selector of every CPU is switched back to "STOP" the STOP LED's of all CPU's indicate a steady light.

By switching from "STOP" to "RUN" the following reactions occur:

- **CPU 920 (M processor):**  
QVZ (timeout) and slowly flashing STOP LED
- **CPU 922 and 921 (R and S processor) and CPU 928:**  
slowly flashing STOP LED

The STOP LED does not remain steady if CPU 921 (S processor) is switched back to "STOP". This can be produced with a cold restart (see above).

**Check:**

*Is the coordinator connected?*

**C ... after step 6**

**Fault/error reaction 1:**

All CPU's remain stopped.

**Check:**

*Are the mode selectors of all CPU's set to "RUN"?*

Delayed starting of individual CPU's is not possible. Switch coordinator back to "STOP". Switch all CPU's to "RUN". Switch coordinator to "RUN" again.

In test mode, only CPU's with their switches set to "RUN" start if the coordinator is switched from "STOP" to "TEST".

**Fault/error reaction 2:**

CPU 921 (S processor):  
STOP LED immediately flashes slowly. This remains so when switching back from "RUN" to "STOP".

CPU 922 (R processor), CPU 928, CPU 946/947:  
STOP LED immediately flashes slowly but changes to steady light when switching back from "RUN" to "STOP".

CPU 920 (M processor):  
The red STOP LED continues to be steady. If the coordinator is switched from "STOP" to "RUN" the STOP LED of CPU 920 (M processor) flashes slowly.

**Check:**

*Was each CPU 921 (S processors) started using cold restart with reset and all other CPU's using cold restart?*

If the coordinator is operating in the test mode, CPU's which were set for a cold restart can be started if the coordinator is switched from "STOP" to "TEST".

**Fault/error reaction 3:**

Using the coordinator C:

if the coordinator C is switched from "STOP" to "RUN" in step 6 all CPU's remain stopped (STOP LED steady). No I stack can be read out at the PG. All CPU's output the BASP signal.

If switch S1.3 on the coordinator C is set, i.e. if the test function is coded, switching the mode selector from "STOP" to "TEST" does not result in a reaction in CPU 921 (S processor), CPU 922 (R processor), CPU 928, and CPU 946/947. The green RUN LED of CPU 920 (M processor), however, stays on (steady). The BASP signal is output by all CPU's. The processor cannot be operated using the front panel switches (nor those of the coordinator).

After switching the power supply of the CC off and back on, the STOP LED's of all CPU's (also CPU 920 if switched to "STOP") are lit steadily.

**Check:**

*Setting of the number of CPU slots occupied on the coordinator C.*

# SIEMENS

## SIMATIC S5

Multiprocessor Communication  
S5-135U, CPU 922 (R Processor),  
CPU 928 and CPU 928B  
S5-155U, CPU 946/947

---

User's Guide

C79000-B8576-C468-05

---

|            | Page   |
|------------|--|
| <b>1</b>   | <b>Introduction ..... 3</b>  |
| <b>1.1</b> | <b>Configuration..... 4</b>  |
| <b>1.2</b> | <b>Principle ..... 4</b>   |
| <b>1.3</b> | <b>Sender/Receiver Identification..... 5</b>   |
| <b>1.4</b> | <b>Buffering the Data ..... 6</b>  |
| <b>1.5</b> | <b>System Restart ..... 9</b>  |
| <b>1.6</b> | <b>Calling and Nesting the Special Function Organization Blocks<br/>OB 200 and OB 202 to OB 205 ..... 10</b> |
| <b>1.7</b> | <b>Parallel Processing in a Multiprocessor Programmable Controller..... 11</b>                               |
| <b>1.8</b> | <b>Required Memory Areas ..... 11</b>  |
| <b>1.9</b> | <b>Runtime..... 12</b>   |
| <b>2</b>   | <b>Parameter Assignment ..... 14</b>   |
| <b>2.1</b> | <b>Evaluating the Output Parameters ..... 14</b>   |
| 2.1.1      | Condition Codes..... 15  |
| 2.1.2      | Condition Code Byte: Initialization Conflict/Error/Warning ..... 16  |
| <b>3</b>   | <b>INITIALIZE Function (OB 200) ..... 21</b>   |
| <b>3.1</b> | <b>Input Parameters ..... 23</b>   |
| 3.1.1      | Mode (Automatic / Manual) ..... 23   |
| 3.1.2      | Number of CPUs ..... 24  |
| 3.1.3      | Block ID and Number / Start Address of the Assignment List..... 24   |
| <b>3.2</b> | <b>Output Parameters ..... 26</b>  |
| 3.2.1      | Condition Code Byte ..... 26   |
| 3.2.2      | Total Capacity ..... 28  |
| <b>4</b>   | <b>SEND Function (OB 202) ..... 29</b>   |
| <b>4.1</b> | <b>Input Parameters ..... 29</b>   |
| 4.1.1      | Receiving CPU..... 29  |
| 4.1.2      | Block ID and Number / Field Number ..... 29  |
| <b>4.2</b> | <b>Output Parameters ..... 31</b>  |
| 4.2.1      | Condition Code Byte ..... 31   |
| 4.2.2      | Transmitting Capacity ..... 33   |

|            |  |           |
|------------|--|-----------|
| <b>5</b>   | <b>SEND TEST Function (OB 203)</b> .....                           | <b>34</b> |
| <b>5.1</b> | <b>Input Parameters</b> .....                                      | <b>34</b> |
| 5.1.1      | Receiving CPU.....   | 34        |
| <b>5.2</b> | <b>Output Parameters</b> .....                                     | <b>34</b> |
| 5.2.1      | Condition Code Byte .....  | 34        |
| 5.2.2      | Transmitting Capacity .....  | 35        |
| <br>       |  |           |
| <b>6</b>   | <b>RECEIVE Function (OB 204)</b> .....                             | <b>36</b> |
| <b>6.1</b> | <b>Input Parameters</b> .....                                      | <b>36</b> |
| 6.1.1      | Transmitting CPU.....  | 36        |
| <b>6.2</b> | <b>Output Parameters</b> .....                                     | <b>37</b> |
| 6.2.1      | Condition Code Byte .....  | 37        |
| 6.2.2      | Receiving Capacity .....   | 38        |
| 6.2.3      | Block ID and Number.....   | 38        |
| 6.2.4      | Address of the First/Last Received Data Word .....                 | 39        |
| <br>       |  |           |
| <b>7</b>   | <b>RECEIVE TEST Function (OB 205)</b> .....                        | <b>40</b> |
| <b>7.1</b> | <b>Input Parameters</b> .....                                      | <b>40</b> |
| 7.1.1      | Transmitting CPU.....  | 40        |
| <b>7.2</b> | <b>Output Parameters</b> .....                                     | <b>40</b> |
| 7.2.1      | Condition Code Byte .....  | 40        |
| 7.2.2      | Receiving Capacity .....   | 41        |
| <br>       |  |           |
| <b>8</b>   | <b>Applications</b> .....  | <b>42</b> |
| <b>8.1</b> | <b>Calling the Special Function OB Using Function Blocks</b> ..... | <b>42</b> |
| 8.1.1      | Setting Up a Buffer (FB 200).....                                  | 43        |
| 8.1.2      | Sending a Block of Data (FB 202) .....                             | 45        |
| 8.1.3      | Testing the Transmitting Capacity (FB 203).....                    | 47        |
| 8.1.4      | Receiving a Block of Data (FB 204).....                            | 48        |
| 8.1.5      | Testing the Receiving Capacity (FB 205) .....                      | 50        |
| <b>8.2</b> | <b>Transferring Data Blocks</b> .....                              | <b>51</b> |
| 8.2.1      | Functional Description.....  | 51        |
| 8.2.2      | Transferring a Data Block (FB 110) .....                           | 51        |
| 8.2.3      | Application Example (for the S5-135U).....                         | 54        |
| <b>8.3</b> | <b>Extending the IPC Flag Area</b> .....                           | <b>56</b> |
| 8.3.1      | The Problem.....   | 56        |
| 8.3.2      | The Solution.....  | 57        |
| 8.3.3      | Data Structure.....  | 57        |
| 8.3.4      | Program Structure.....   | 60        |
| 8.3.5      | Sending Data Word Areas (FB 100) .....                             | 62        |
| 8.3.6      | Receive Data Word Areas (FB 101) .....                             | 65        |
| 8.3.7      | Application Example (for S5-135U).....                             | 68        |

# 1 Introduction

You can operate the multiprocessor programmable controllers S5-135U and S5-155U with up to four CPUs. You can use the following "tools" individually or in combination to exchange data between the CPUs:

- F flags are transferred, if you define them as interprocessor communication (IPC) output flags in **one** CPU and as IPC input flags in one or more CPUs.
- To transfer data blocks, or to be more precise, blocks of data with a maximum length of 64 bytes (= 32 data words), you can use the following special functions that are integrated in the CPU:

|                               |                         |
|-------------------------------|-------------------------|
| <b>INITIALIZE (OB 200) :</b>  | preassign               |
| <b>SEND (OB 202):</b>         | send a block of data    |
| <b>SEND TEST (OB 203):</b>    | test sending capacity   |
| <b>RECEIVE (OB 204):</b>      | receive a block of data |
| <b>RECEIVE TEST (OB 205):</b> | test receiving capacity |

To use these functions, you only require basic knowledge of the STEP 5 programming language and the way in which SIMATIC S5 programmable controllers operate. You can obtain this basic information from the publications listed in the table of documentation.

Whereas the IPC flags are updated "automatically" by the system program, you must call the **INITIALIZE, SEND, SEND TEST, RECEIVE and RECEIVE TEST** functions as special function organization blocks using the **JU OB** or **JC OB** operations.



## 1.1 Configuration

### S5-135U or S5-155U

These PLCs contain the S5 bus and in the multiprocessor mode they also have the following components:

- **1 coordinator 923C**

This module contains four pages. These are memory areas of 1024 bytes. They all occupy the address area F400H to F7FFH. You select (address) the "current" page using the select register (also known as the identification or page address register, similar to chip select). The select numbers 252, 253, 254 and 255 are **fixed** as the four pages of the 923C coordinator and are used for multiprocessor communication.

- **2 to 4 CPUs**

For the S5-135U: CPU 922 (R processor), CPU 928 , CPU 928B or CPU 920 (M processor)

For the S5-155U: CPU 946/947, CPU 922 (R processor), CPU 928 , CPU 928B or CPU 920 (M processor).

These CPUs can exchange data with each other in any combination, you can also use "handling blocks" which also work with page addressing without any restrictions.

If you have one or more additional CPU 921s (S processors) in the same rack, they **cannot** take part in the multiprocessor communication. You must not call the S processor handling blocks as long as R and M processors, CPU 928s, CPU 928Bs and CPU 946/947s are processing their handling blocks or are involved in multiprocessor communication. CPUs can, however, always communicate via IPC flags.

## 1.2 Principle

To transfer data, you must activate the SEND function on the transmitting CPU and the RECEIVE function on the receiving CPU.

The data words of a DB or DX data block located in the transmitting CPU are transported via the coordinator 923C to the receiving CPU one after the other and written to the DB or DX data block with the same number and under the same data word address; i.e. this represents a "1:1" copying.

### Example

|                   | Data to be sent in the transmitting CPU | Data received in the receiving CPU |
|-------------------|---|------------------------------------|
| Data block        | DB 17                                   | DB 17                              |
| Data word address | DW 32 to DW 63                          | DW 32 to DW 63                     |

The amount of data that can be transferred with the SEND and RECEIVE functions is normally 32 words.

If the block length (without header) is not a multiple of 32 words, the last block of data to be transferred is an exception and is less than 32 words.

The data block in the receiving CPU can be longer or shorter than the data block to be sent. It is, however, important that the data words transferred by the SEND function exist in the receiving block; otherwise the RECEIVE function signals an error.

### 1.3 Sender/Receiver Identification

The CPUs are numbered so that the leftmost CPU has the number 1 and each subsequent CPU to the right has a number increased by 1.

#### Example

S5-135U/155U:

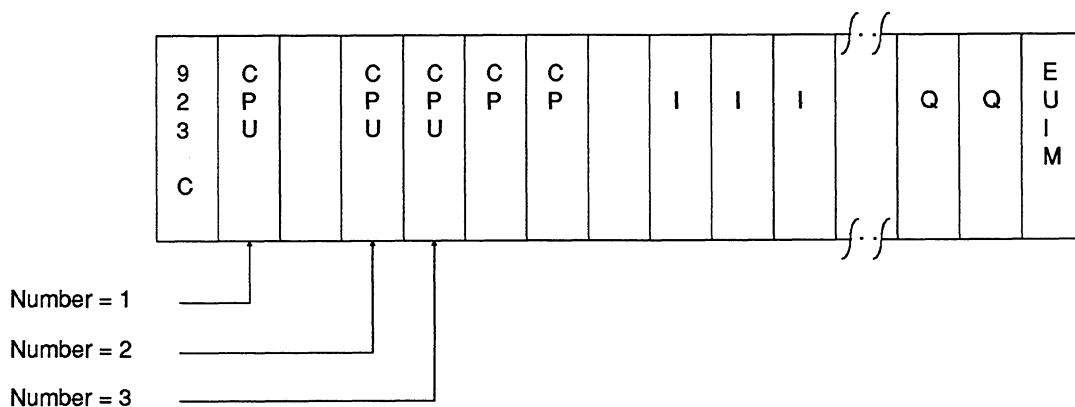
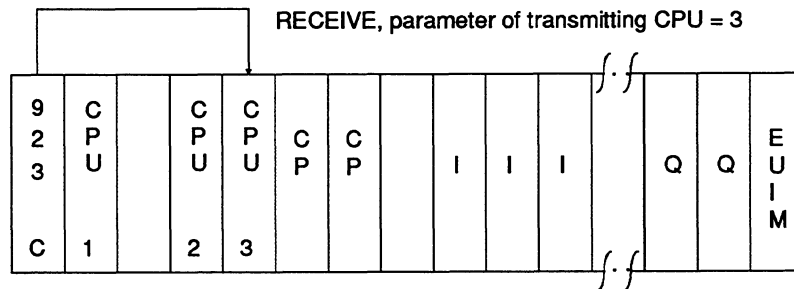


Fig. 1 Sender/receiver identification



## 2nd step



- 1st step the buffer is based on the FIFO principle (first in, first out queue principle). The data is received in the order in which it is sent. This applies to each individual transmission (identified by the transmitting and receiving CPU) and is independent of other connections.
- 2nd step the buffer is battery-backed; this means that the "automatic warm restart following power down" is possible without any restrictions. A loss of power during a data transfer does not cause any loss of data in the programmable controller.

The coordinator 923C has a memory capacity of 48 data fields each capable of containing 32 words. The INITIALIZE function assigns these fields to individual connections.

Each **memory field** (always with a length of 32 words) can hold exactly one **block of data** (with a length between 1 data word and 32 data words). The SEND block enters one **block of data** in a **memory field** from where it is read out by the RECEIVE block.

The number of memory fields assigned to a connection is directly related to the parameters for the transmitting capacity (SEND, SEND TEST function) and receiving capacity (RECEIVE, RECEIVE TEST function).

The transmitting capacity indicates how many of the memory fields reserved for a connection are free at any particular time.

The receiving capacity indicates how many of the memory fields reserved for a connection are occupied at any particular time.


The sum of the transmitting and receiving capacity parameters is always equal to the number of memory fields reserved for a connection.

**Example**

The following table indicates a possible data transfer sequence assuming that the connection "from CPU 3 to CPU 2" has seven memory fields assigned by the INITIALIZE function.

| Transmitter: CPU 3                  |  | Receiver: CPU 2                             |          |  |
|-------------------------------------|--|---|----------|--|
| Sequence                            | Transmitting capacity (free memory fields) | Receiving capacity (occupied memory fields) | Sequence |  |
|                                     |  | Time<br>↓                                   |          |  |
| Sends a blocks of data (A)          | 7  |   | 0        | Initialize                               |
| Sends four blocks of data (B,C,D,E) | 6  |   | 1        |  |
|                                     | 2  |   | 5        | Receives two blocks of data (A, B)       |
| Sends four blocks of data (F,G,H,I) | 4  |   | 3        |  |
|                                     | 0  |   | 7        | Receives five blocks of data (C,D,E,F,G) |
| Sends two blocks of data (K,L)      | 5  |   | 2        | Receives two blocks of data (H,I)        |
|                                     | 5  |   | 2        |  |

**REMEMBER**

 Sending/receiving n data blocks means that the corresponding function is called n times.

To simplify the representation, at any one time, data can either be sent or received in this example. It is, however, possible and useful to transmit (CPU 3) and receive (CPU 2) simultaneously (see "Parallel processing in a multiprocessor programmable controller"). In the example, blocks H and I are received while blocks K and L are sent.

The example illustrates the queue organization of the buffer; the blocks of data sent first (A,B,C...) are received first (A,B,C...).

## Summary

Buffering data on the coordinator 923C allows the asynchronous operation of transmitting and receiving CPUs and compensates for their different processing speeds.

Since the capacity of the buffer is limited, the receiver should check "often" and "regularly" whether there are data in the buffer (RECEIVE TEST function, receiving capacity > 0) and should attempt to fetch stored data (RECEIVE function). Ideally, the RECEIVE function should be repeated until the receiving capacity is zero. This means that the transmitted data are not buffered for a longer period of time and that the receiver always has the current data. This also means that memory fields remain free (the transmitting capacity is increased) and prevents the sender from being blocked (i.e. when the transmitting capacity is zero).

A receiving capacity of zero represents the ideal state (i.e. all transmitted data have been fetched by the receiver), on the other hand a transmitting capacity of zero indicates incorrect planning, as follows:

- the SEND function is called too often.
- the RECEIVE function is not called often enough,
- there are not enough memory fields assigned to the connection. The capacity of the buffer is insufficient to compensate temporary imbalances in the frequency with which the CPUs transmit and receive data.

## 1.5 System Restart

If you require multiprocessor communication, then all the CPUs involved must go through the **same** STOP-RUN transition (= RESTART), i.e. all the CPUs go through a COLD RESTART or all CPUs go through a WARM RESTART.

You must make sure that the restart of at least all the CPUs involved in the communication is **uniform** (see Chapter 10), in the following ways:

- **direct operation** (front switch, programmer),
- **parameter assignment** (DX 0) and/or
- **programming** (using the special function organization block OB 223 "stop if non-uniform restarts occur in the multiprocessor mode").

## COLD RESTART

In organization block OB 20 (COLD RESTART) **one** CPU must set up the buffer (in the 923C) using the INITIALIZE function. Any existing data is lost.

Following this, i.e. during the RESTART, you can call the SEND, SEND TEST, RECEIVE, RECEIVE TEST functions in the individual CPUs. With appropriate programming, you must make sure that this only occurs after the buffer in the coordinator has been correctly initialized.

On completion of the RESTART, i.e. in the RUN mode, the user program is processed from the beginning, i.e. from the first operation in OB 1 or FB 0.

## WARM RESTART

You must **not** use the INITIALIZE function in the organization blocks OB 21 (MANUAL WARM RESTART) and OB 22 (AUTOMATIC WARM RESTART). Calling the SEND, SEND TEST, RECEIVE, RECEIVE TEST functions can cause problems (refer to the following section).

On completion of the WARM RESTART, i.e. in the RUN mode, the user program is not processed from the start, **but** from the point at which it was interrupted. The point of interruption can, for example, be within the SEND function.

## 1.6 Calling and Nesting the Special Function Organization Blocks OB 200 and OB 202 to OB 205

The simplest procedure is as follows:

- program the call for the INITIALIZE function only in the cold restart organization block OB 20;
- program the call for the SEND, SEND TEST, RECEIVE, RECEIVE TEST functions either **only** within the cyclic program **or only** within the time-driven program.

### REMEMBER



Depending on the assignment of parameters in DX 0 ("interrupts at command boundaries" for the CPU 928B, CPU 928 and CPU 920, or "155U mode" for the CPU 946/947), and the type of program execution (WARM RESTART, interrupt handling, e.g. OB 26 for cycle time error) it is possible that one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE and RECEIVE TEST can be interrupted. If a user interface inserted at the point of interruption (e.g. OB 13 when interrupts are possible at operation boundaries or OB 22 following power down) also contains one of the functions SEND, SEND TEST, RECEIVE and RECEIVE TEST an illegal call (double call) is recognized and an error is signalled (error number 67, Section 2.1.2).

## 1.7 Parallel Processing in a Multiprocessor Programmable Controller

Once you have completed the assignment of the buffer (INITIALIZE function), you can execute the functions SEND, SEND TEST, RECEIVE and RECEIVE TEST in any combination and with any parameter assignment in all the CPUs simultaneously and parallel to each other.

Taking a single connection (from CPU 'SE' to CPU 'RE') it is possible to execute the SEND function (CPU 'SE') and the RECEIVE function (CPU 'RE') simultaneously. While CPU 'SE' is sending blocks of data to the coordinator, CPU 'RE' can receive (fetch) buffered blocks of data from the coordinator.

## 1.8 Required Memory Areas

The special function organization blocks OB 200 and OB 202 to OB 205 do not require a working area (e.g. for buffering variables) and do not call data blocks. They do, of course, access areas containing parameters, although only the parameters marked as output parameters are modified. These OBs also affect the condition codes (CC1, RLO etc., see Section 2.1).

- CPU 922, CPU 928, CPU 928B: The contents of ACCU 1 to ACCU 4 and the contents of the registers are not affected by the special function OBs for multiprocessor communication.
- CPU 946/947: The contents of all registers and ACCU 1, 2 and 3 remain the same, only the contents of ACCU 4 are affected.



## 1.9 Runtime

| Special function OB   |   | Runtime   |   |  |  |
|-----------------------|---|---|---|--|--|
| Block / name          | Block function                          | R processor   | CPU 928   | CPU 946/947  | CPU 928B   |
| OB 200 / initialize   | Assign buffer                           | 230 ms  | 130 ms  | 128 ms   | 130 ms   |
| OB 202 / send         | Send a block of data (32 data words)    | 806 $\mu$ s (294 $\mu$ s basic time + 16 $\mu$ s / word); 118 $\mu$ s if a warning occurs | 666 $\mu$ s (250 $\mu$ s basic time + 13 $\mu$ s / word); 115 $\mu$ s if a warning occurs | 762 $\mu$ s (426 $\mu$ s basic time + 21 $\mu$ s / double word); 243 $\mu$ s if a warning occurs | 696 $\mu$ s (280 $\mu$ s) basic time + 31 $\mu$ s / word); 145 $\mu$ s if a warning occurs |
| OB 203 / send test    | Test transmitting capacity              | 72 $\mu$ s  | 50 $\mu$ s  | 207 $\mu$ s  | 80 $\mu$ s   |
| OB 204 / receive      | Receive a block of data (32 data words) | 825 $\mu$ s (281 $\mu$ s basic time + 17 $\mu$ s / word); 115 $\mu$ s if a warning occurs | 660 $\mu$ s (244 $\mu$ s basic time + 13 $\mu$ s / word); 98 $\mu$ s if a warning occurs  | 772 $\mu$ s (421 $\mu$ s basic time + 22 $\mu$ s / double word); 243 $\mu$ s if a warning occurs | 690 $\mu$ s (274 $\mu$ s basic time + 13 $\mu$ s / word); 128 $\mu$ s if a warning occurs  |
| OB 205 / receive test | Test receiving capacity                 | 70 $\mu$ s  | 48 $\mu$ s  | 223 $\mu$ s  | 78 $\mu$ s   |

The "runtime" is the processing time of the special function organization blocks; the time from calling a block to its termination can be much greater if it is interrupted by higher priority activities (e.g. updating timers, processing closed loop controllers etc.).

The runtimes listed above assume that of four CPUs inserted in a rack, only the CPU whose runtimes are being measured accesses the SIMATIC S5 bus. If other CPUs use the bus intensively, the runtime increases particularly for the send/receive functions.

An important factor of a connection (from CPU 'SE' to CPU 'RE') is the total data transfer time. This is made up of the following components:

- time required to send (see runtime)
- length of time the data are buffered (on the 923C coordinator)
- the time required to receive data (see runtime)

**The length of time that the data are "in transit" is largely dependent on the length of time that the data is buffered and therefore on the structure of the user program (see "Buffering Data").**

## 2 Parameter Assignment

The "actual" parameters are located in a maximum 10 byte long data field in the F flag area. The number of the first flag byte in the data field (= pointer to the data field) must be loaded in ACCU-1-L. Permitted values are 0 to 246.

The data field is divided into an area for **input parameters** and an area for **output parameters**.

- **Input parameters**

All or part of the input parameters are read and evaluated by the functions, the functions do not write to this area.

- **Output parameters**

Some or all of the output parameters are written to by the functions, the functions do not read this area.

### REMEMBER



You can assign a **flag area with 10 flag bytes** for all communications functions. The functions themselves require different numbers of bytes. Refer to the description of the single functions (Chapters 3 to 7).

#### Example: data field with parameters for the RECEIVE function (OB 204)

|           |                      |                  |
|-----------|----------------------|------------------|
| FY x + 0: | transmitting CPU     | input parameter  |
| FY x + 1: | —                    | not used         |
| FY x + 2: | condition code byte  | output parameter |
| FY x + 3: | receiving capacity   | output parameter |
| FY x + 4: | block ID             | output parameter |
| FY x + 5: | block number         | output parameter |
| FY x + 6: | address of the first | output parameter |
| FY x + 7: | received data word   | output parameter |
| FY x + 8: | address of the last  | output parameter |
| FY x + 9: | received data word   | output parameter |

This example illustrates that the number of the first F flag byte in the data field must not be higher than FY 246, since otherwise the parameter field of up to 10 bytes would exceed the limits of the flag area (FY 255).

### 2.1 Evaluating the Output Parameters

Output parameters are data made available to the user program for evaluation. Among other things, they indicate whether or not a function could be executed and if not they indicate the reason for the termination of the function.

### 2.1.1 Condition Codes

The INITIALIZE, SEND, SEND TEST, RECEIVE and RECEIVE TEST functions affect the condition codes (see programming instructions for your CPUs, general notes on the STEP 5 operations):

- the OV and OS bits (word condition codes) are always cleared,
- the OR, STA, ERAB bits (bit condition codes) are always cleared,
- RLO, CC 0 and CC 1 indicate whether a function has been executed correctly and completely.

RLO = 0: Function executed correctly and completely

RLO = 1: Function aborted: the pointer to the data field in the flag area may have an illegal value, i.e. the low word of the ACCU contains a value greater than 246.

**In the following sections, it is assumed that the pointer to the data field contains a correct value.** The first byte of the output parameter provides detailed information about the cause of termination.

CC 1 = 1: Additional warning information (warning number 1 or 2)

CC 0 = 1: Additional error indication (error number 1-9)

| Situation  | Condition codes |      |      | Evaluation with operation |
|--|-----------------|------|------|---------------------------|
|  | RLO             | CC 1 | CC 0 |                           |
| Function executed completely and correctly           | 0               | 0    | 0    | JC =                      |
| Function aborted, pointer to data field illegal      | 1               | 0    | 0    | JC =                      |
| Function aborted owing to an initialization conflict | 1               | 0    | 0    | JC =                      |
| Function aborted owing to an error                   | 1               | 0    | 1    | JC = and JM =             |
| Function aborted owing to a warning                  | 1               | 1    | 0    | JC = and JP =             |

### 2.1.2 Condition Code Byte: Initialization Conflict/Error/Warning

The first byte in the field of the output parameters (condition code byte) also indicates whether or not a function has been correctly and completely executed. This byte contains detailed information about the cause of termination of a function.

Assuming that at least the pointer to the data field contains a correct value, this byte is **always** relevant.

If the function has been executed correctly and completely, all the bits are cleared (= 0), and all other output parameters are relevant.

If the function is aborted with a warning (bit  $2^7 = 1$ ), only the condition code for the transmitting/receiving capacity is relevant, other output parameters (if they exist) are unchanged.

If the function is aborted owing to an error (bit  $2^6 = 1$ ) or an initialization conflict (bit  $2^5 = 1$ ), all other output parameters remain unchanged.

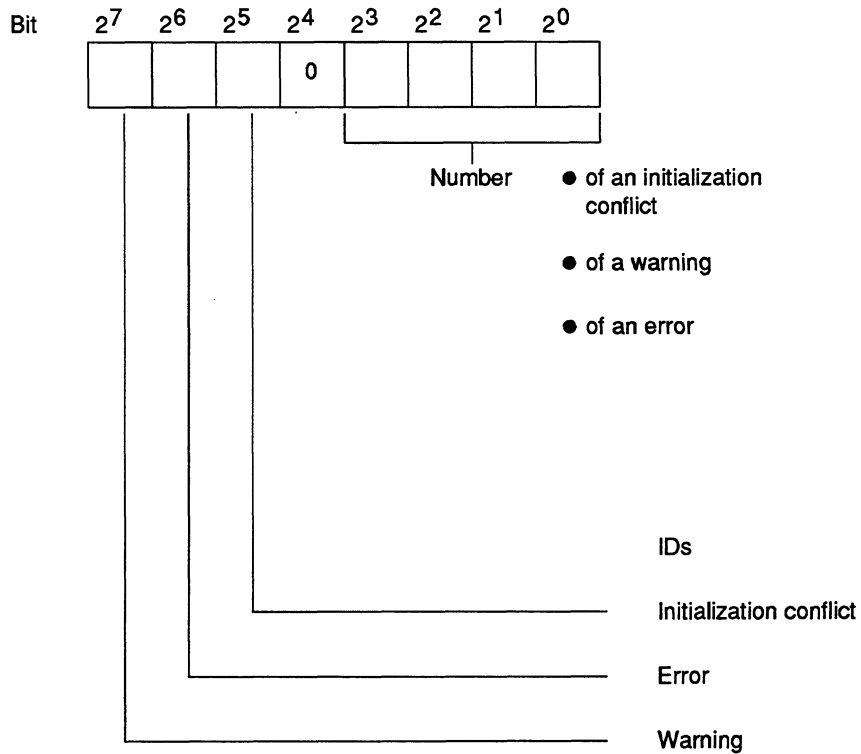


Fig. 2 Coding of the first byte

## Evaluation

The identifiers in bit positions  $2^5$  to  $2^7$  indicate the significance of the numbers in bit positions  $2^0$  to  $2^3$ .

Apart from this bit-by-bit evaluation, it is also possible to interpret the whole condition code byte as a fixed point number without sign. If you interpret the condition code byte as a **byte**, the groups of numbers have the following significance:

| Number group | Significance   |
|--------------|--|
| 0            | Function executed correctly and completely           |
| 33 to 42     | Function aborted owing to an initialization conflict |
| 65 to 73     | Function aborted owing to an error                   |
| 129 to 130   | Function aborted owing to a warning                  |

The values of the following numbers **also** indicate **the order** in which errors or initialization conflicts were recognized and indicated by the functions.

### Example

The SEND function indicates an error and is not executed. If you then make program and/or parameter modifications and the SEND function once again indicates an error with a higher number than previously, you can assume that you have corrected one of several errors.

### Initialization conflict

An initialization conflict can only occur with the INITIALIZATION function. If a conflict occurs, you must modify the program or parameters.

### Initialization conflict numbers (evaluation of the condition code byte as a byte)

- (33) The pages required for multiprocessor communication (numbers 252 to 255) are not or not all available.
- (34) The pages required for multiprocessor communication (numbers 252 to 255) are defective.
- (35) The parameter "automatic/manual" is illegal. The following errors are possible:

The "automatic/manual" ID is less than 1.

The "automatic/manual" ID is greater than 2.

- **(36)** The parameter "number of CPUs" is illegal. The following errors are possible:
  - The number of CPUs is less than 2.
  - The number of CPUs is greater than 4.
  
- **(37)** The parameter "block ID" is illegal. The following errors are possible:
  - The block ID is less than 1.
  - The block ID is greater than 2.
  
- **(38)** The parameter "block number" is illegal, since it is a data block with a special significance. The following errors are possible:
  - If block ID = 1 : DB 0, DB 1, DB 2
  - If block ID = 2 : DX 0, DX 1, DX 2
  
- **(39)** The parameter "block number" is incorrect, since the data block does not exist.
  
- **(40)** The parameter "start address of the assignment list" is too high or the data block is too short.
  
- **(41)** The assignment list in the data block is not correctly structured.
  
- **(42)** The sum of the assigned memory fields is greater than 48.

## Errors

If an error occurs, you must change the program/parameters.

### Error numbers (evaluation of the condition code byte as a byte)

- **(65)** The parameter "receiving CPU" (SEND, SEND TEST) is illegal, since it is a data block with a special significance. The following errors are possible:
  - The number of the receiving CPU is greater than 4.
  - The number of the receiving CPU is less than 1.
  - The number of the receiving CPU is the same as the CPU's own number.
  
- **(66)** The parameter "transmitting CPU" (RECEIVE, RECEIVE TEST) is illegal, since it is a data block with a special significance. The following errors are possible:
  - The number of the transmitting CPU is greater than 4.
  - The number of the transmitting CPU is less than 1.
  - The number of the transmitting CPU is the same as the CPU's own number.
  
- **(67)** The special function organization block call is wrong (SEND, RECEIVE, SEND TEST, RECEIVE TEST). The following errors are possible:
  - a) Secondary error, since the INITIALIZE function could not be called or was terminated by an initialization conflict.
  - b) Double call: the call for this function, SEND, SEND TEST, RECEIVE or RECEIVE TEST is illegal, since one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE or RECEIVE TEST has already been called in this CPU in a lower processing level (e.g. cyclic program execution). (See "Calling and nesting the special function organization blocks".)
  - c) The CPU's own number is incorrect (system data corrupted); following power down/power up the CPU number is generated again by the system program.
  
- **(68)** The management data (queue management) of the selected connections are incorrect; set up the buffer in the coordinator 923C again using the INITIALIZE function (SEND, RECEIVE, SEND TEST, RECEIVE TEST).



- (69) The parameter "block ID" (SEND) or the block ID provided by the sender (RECEIVE) is illegal. The following errors are possible:
  - The block ID is less than 1.
  - The block ID is greater than 2.
  
- (70) The parameter "block number" (SEND) or the block number supplied by the sender (RECEIVE) is illegal, since it is a data block with a special significance. The following errors are possible:
  - If the block ID = 1 : DB 0, DB 1, DB 2
  - If the block ID = 2 : DX 0, DX 1, DX 2
  
- (71) The parameter "block number" (SEND) or the block number provided by the sender (RECEIVE) is incorrect. The specified data block does not exist.
  
- (72) The parameter "field number" (SEND) is incorrect. The data block is too short or the field number too high.
  
- (73) The data block is not large enough to receive the block of data transmitted by the sender (RECEIVE).

### **Warning**

The function could not be executed; the function call must be repeated, e.g. in the next cycle.

### **Warning numbers (evaluation of the condition code byte as a byte)**

- (129) The SEND function cannot transfer data, since the transmitting capacity was already zero when the function was called.
  
- (130) The RECEIVE function cannot accept data, since the receiving capacity was already zero when the function was called.

### 3 INITIALIZE Function (OB 200)

#### Call parameters

##### 1st data field

Before calling OB 200, you must supply the input parameters in the data field. OB 200 requires eight F flag bytes in the data field for input and output parameters:

|           |   |                  |
|-----------|---|------------------|
| FY x + 0: | Mode (automatic/<br>manual)             | input parameter  |
| FY x + 1: | Number of CPUs                          | input parameter  |
| FY x + 2: | Block ID                                | input parameter  |
| FY x + 3: | Block number                            | input parameter  |
| FY x + 4: | Start address of the<br>assignment list | input parameter  |
| FY x + 5: |   | input parameter  |
| FY x + 6: | Condition code byte                     | output parameter |
| FY x + 7: | Total capacity                          | output parameter |

##### 2 ACCU-1-L:

No. of the 1st flag byte "x" in the data field,  
permitted values:

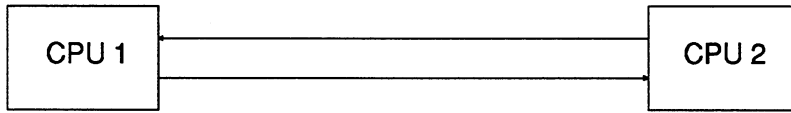
ACCU-1-LH: 0  
ACCU-1-LL: 0 to 246

To transfer data from one CPU to another CPU, the data must be temporarily buffered. The INITIALIZE function sets up a buffer on the KOR 923C coordinator.

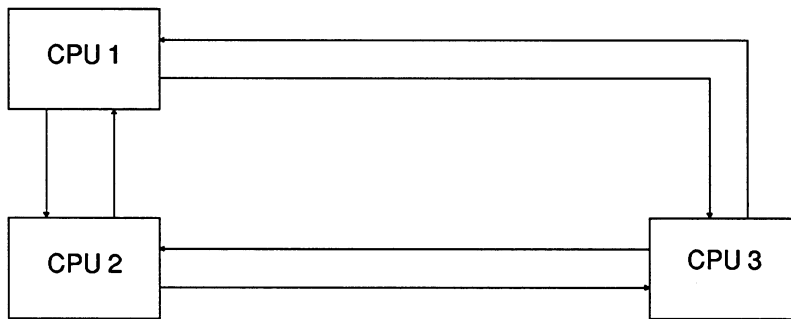
The memory capacity is stipulated in fields (with a length of 32 words).

Each memory field (always with a length of 32 words) accepts one block of data (with a length between one data word and 32 data words). A block of data is entered in a memory field by a SEND block and read out by a RECEIVE block.

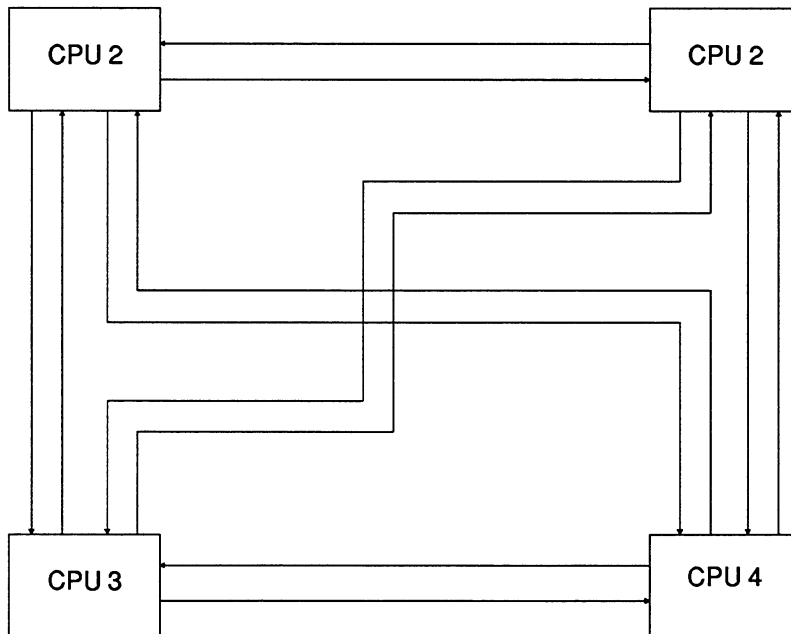
If you are using two CPUs, there are two connections (transfer directions, "channels"):




If you are using three CPUs, there are six connections:



If you are using four CPUs, there are twelve connections:



The INITIALIZE function specifies how the total of 48 available memory fields are assigned to the maximum twelve connections. This means that each possible connection, specified by the parameters "transmitting CPU" and "receiving CPU" has a certain memory capacity available.

|   |  |
|---|--|
|  | <p><b>REMEMBER</b></p> <p>Before you can call the SEND / RECEIVE / SEND TEST / RECEIVE TEST functions, one CPU must have already called the INITIALIZE function and executed it completely and without errors.</p> |
|---|--|

If the INITIALIZE function is called several times, one after the other, the last assignment made is valid. While a CPU is processing the INITIALIZATION function, no other functions including the INITIALIZE function can be called on other CPUs.

### 3.1 Input Parameters

#### 3.1.1 Mode (Automatic / Manual)

Mode = 1 :                    automatic  
 Mode = 2 :                    manual  
 Mode = 0 or 3 to 255 :    illegal, causes an initialization conflict

#### "Automatic" mode

If you select the "automatic" mode, the memory fields available are divided **equally** according to the number of CPUs:

| Number of CPUs | Number of connections                      | Memory fields per connection |
|----------------|--|------------------------------|
| 2              | 2  | 24                           |
| 3              | 6  | 8                            |
| 4              | 12   | 4                            |
| 0; 1; 5 to 255 | Illegal: causes an initialization conflict |                              |

## "Manual" mode

If you select the "manual" mode, you must create an assignment list in a data block in which the 48 (or less) available memory fields are assigned to the maximum 12 connections according to a fixed scheme. This function is particularly useful when some connections have far more data traffic than others. For example, CPUs 921 (S processors) cannot take part in the multiprocessor communication described here; the potential connections between this CPU and other CPUs do not therefore need memory fields and should not have memory fields assigned to them. The parameters

- block ID,
- block number and the
- start address of the assignment list

specify where the assignment list is stored. These three parameters are therefore only relevant for the "manual" mode.

### 3.1.2 Number of CPUs

This parameter is only relevant if you select the "automatic" mode; (see 3.1.1)

### 3.1.3 Block ID and Number / Start Address of the Assignment List

These parameters are only relevant if you select the "manual" mode.

#### Block ID and number

|                      |  |
|----------------------|--|
| ID = 1:              | DB data block                              |
| ID = 2:              | DX data block                              |
| ID = 0 or 3 to 255 : | illegal, causes an initialization conflict |

For the block number, you specify the number of the DB or DX data block in which the assignment list is stored.

#### Start address of the assignment list

Along with the block ID and number, this specifies the area (or more precisely, the start address of the area) in which the assignment list is stored.

The assignment list contains further input parameters for the INITIALIZE function, i.e. this area is only read (the contents are not changed). The assignment list has the structure shown on the following page:

## Assignment list

| Data word | : | Format | Value       | : | Significance        |
|-----------|---|--------|-------------|---|---------------------|
| DW n+ 0   | : | KS     | S1          | : | Transmitter = CPU 1 |
| DW n+ 1   | : | KY     | 2, <b>a</b> | : | Receiver = CPU 2    |
| DW n+ 2   | : | KY     | 3, <b>b</b> | : | Receiver = CPU 3    |
| DW n+ 3   | : | KY     | 4, <b>c</b> | : | Receiver = CPU 4    |
| DW n+ 4   | : | KS     | S2          | : | Transmitter = CPU 2 |
| DW n+ 5   | : | KY     | 1, <b>d</b> | : | Receiver = CPU 1    |
| DW n+ 6   | : | KY     | 3, <b>e</b> | : | Receiver = CPU 3    |
| DW n+ 7   | : | KY     | 4, <b>f</b> | : | Receiver = CPU 4    |
| DW n+ 8   | : | KS     | S3          | : | Transmitter = CPU 3 |
| DW n+ 9   | : | KY     | 1, <b>g</b> | : | Receiver = CPU 1    |
| DW n+ 10  | : | KY     | 2, <b>h</b> | : | Receiver = CPU 2    |
| DW n+ 11  | : | KY     | 4, <b>i</b> | : | Receiver = CPU 4    |
| DW n+ 12  | : | KS     | S4          | : | Transmitter = CPU 4 |
| DW n+ 13  | : | KY     | 1, <b>k</b> | : | Receiver = CPU 1    |
| DW n+ 14  | : | KY     | 2, <b>l</b> | : | Receiver = CPU 2    |
| DW n+ 15  | : | KY     | 3, <b>m</b> | : | Receiver = CPU 3    |

**REMEMBER**

You must keep to this structure even if you have less than four CPUs.

The lower case letters a to m in bold face represent numbers between 0 and 48; **the sum of these numbers must not exceed 48.**

The next page shows an example of a completed assignment list.

**Example**

You have three CPUs in your rack, CPU 2 sends a lot of data to the other two CPUs. The other two CPUs, however, only send a small amount of data back to CPU 2 as acknowledgements in a logical handshake. There is no data exchange between CPU 1 and CPU 3.

**Assignment list**

| Data word | : | Format | Value | : | Significance        |
|-----------|---|--------|-------|---|---------------------|
| DW n + 0  | : | KS     | S1    | : | Transmitter = CPU 1 |
| DW n + 1  | : | KY     | 2, 2  | : | Receiver = CPU 2    |
| DW n + 2  | : | KY     | 3, 0  | : | Receiver = CPU 3    |
| DW n + 3  | : | KY     | 4, 0  | : | Receiver = CPU 4    |
| DW n + 4  | : | KS     | S2    | : | Transmitter = CPU 2 |
| DW n + 5  | : | KY     | 1, 22 | : | Receiver = CPU 1    |
| DW n + 6  | : | KY     | 3, 22 | : | Receiver = CPU 3    |
| DW n + 7  | : | KY     | 4, 0  | : | Receiver = CPU 4    |
| DW n + 8  | : | KS     | S3    | : | Transmitter = CPU 3 |
| DW n + 9  | : | KY     | 1, 0  | : | Receiver = CPU 1    |
| DW n + 10 | : | KY     | 2, 2  | : | Receiver = CPU 2    |
| DW n + 11 | : | KY     | 4, 0  | : | Receiver = CPU 4    |
| DW n + 12 | : | KS     | S4    | : | Transmitter = CPU 4 |
| DW n + 13 | : | KY     | 1, 0  | : | Receiver = CPU 1    |
| DW n + 14 | : | KY     | 2, 0  | : | Receiver = CPU 2    |
| DW n + 15 | : | KY     | 3, 0  | : | Receiver = CPU 3    |

**3.2 Output Parameters**

**3.2.1 Condition Code Byte**

This byte informs you whether the INITIALIZE function was executed correctly and completely.

**Initialization conflict**

The initialization conflicts listed are recognized and indicated by the function in the ascending order of their numbers.

If an initialization conflict occurs, you must change the program / parameters.

**Initialization conflict numbers (evaluation of the condition code byte as a byte)**

- (33) The pages required for multiprocessor communication (numbers 252 to 255) are not or not all available.
- (34) The pages required for multiprocessor communication (numbers 252 to 255) are defective.
- (35) The parameter "automatic/manual" is illegal. The following errors are possible:
  - The "automatic/manual" ID is less than 1.
  - The "automatic/manual" ID is greater than 2.
- (36) The parameter "number of CPUs" is illegal. The following errors are possible:
  - The number of CPUs is less than 2.
  - The number of CPUs is greater than 4.
- (37) The parameter "block ID" is illegal. The following errors are possible:
  - The block ID is less than 1.
  - The block ID is greater than 2.
- (38) The parameter "block number" is illegal, since it is a data block with a special significance. The following errors are possible:
  - If block ID = 1 : DB 0, DB 1, DB 2
  - If block ID = 2 : DX 0, DX 1, DX 2
- (39) The parameter "block number" is incorrect, since the data block does not exist.
- (40) The parameter "start address of the assignment list" is too high or the data block is too short.
- (41) The assignment list in the data block is not correctly structured.
- (42) The sum of the assigned memory fields is greater than 48.



### **Errors**

The "error" number group cannot occur with the INITIALIZE function.

### **Warning**

The "warning" number group cannot occur with the INITIALIZE function.

## **3.2.2 Total Capacity**

This parameter specifies how many of the 48 available memory fields are assigned to connections.

In the "automatic" mode, this parameter always has the value 48. In the "manual" mode, it can have a value less than 48. This means that existing memory capacity is not used.

## 4 SEND Function (OB 202)

### Call parameters

#### 1st data field:

Before calling OB 202 you must specify the input parameters in the data field. OB 202 requires six F flag bytes in the data field for input and output parameters:

|           |                       |                  |
|-----------|-----------------------|------------------|
| FY x + 0: | receiving CPU         | input parameter  |
| FY x + 1: | block ID              | input parameter  |
| FY x + 2: | block number          | input parameter  |
| FY x + 3: | field number          | input parameter  |
| FY x + 4: | condition code byte   | output parameter |
| FY x + 5: | transmitting capacity | output parameter |

#### 2. ACCU-1-L:

No. of the first flag byte "x" in the data field:

permitted values: ACCU-1-LH: 0  
ACCU-1-LL: 0 to 246

The SEND function transfers a data block to the buffer of the 923C coordinator. It also indicates how many blocks of data can still be sent and buffered.

### 4.1 Input Parameters

#### 4.1.1 Receiving CPU

The data to be sent are intended for the receiving CPU; the permitted value is between 1 and 4 but must be different from the CPU's own number.

#### 4.1.2 Block ID and Number / Field Number

##### Block ID

|                     |                                  |
|---------------------|----------------------------------|
| ID = 1:             | DB data block                    |
| ID = 2:             | DX data block                    |
| ID = 0 or 3 to 255: | illegal, causes an error message |

**Block number**

The block number, along with the block ID (see above) and the field number (see below) specifies the area from which the data to be sent is taken (and where it is to be stored in the receiving CPU).

Remember that certain data blocks have a special significance, for example, DB 0, DB 1 or DX 0 (see programming instructions for your CPUs). These data blocks must therefore not be used for the data transfer described here.

If you attempt to use these block numbers, the function is aborted with an error message.

**Field number**

The field number indicates the area in which the data to be sent is located.

| Field number | Data area       |                |
|--------------|-----------------|----------------|
|              | First data word | Last data word |
| 0            | DW 0            | DW 31          |
| 1            | DW 32           | DW 63          |
| 2            | DW 64           | DW 95          |
| 3            | DW 96           | DW 127         |
| 4            | DW 128          | DW 159         |
| 5            | DW 160          | DW 191         |
| 6            | DW 192          | DW 223         |
| 7            | DW 224          | DW 255         |
| 8            | DW 256          | DW 287         |
| 9            | DW 288          | DW 319         |
| :            | :               | :              |
| :            | :               | :              |

The following situations are possible:

- If the data block is sufficiently long, you obtain a 32-word long area as shown in the table above.
- If the end of the data block is within the selected field, an area with a length between 1 and 32 words will be transferred.
- If the first data word address is not within the length of the data block, the SEND function detects and indicates an error.

**Example**

Data block with a length of 80 words: DW 0 to DW 74, 5 words are required for the block header.

| Field number | Data area                      |                | Length   |
|--------------|--------------------------------|----------------|----------|
|              | First data word                | Last data word |          |
| 0            | DW 0                           | DW 31          | 32 words |
| 1            | DW 32                          | DW 63          | 32 words |
| 2            | DW 64                          | DW 74          | 11 words |
| 3 and higher | Incorrect parameter assignment |                |          |

**4.2 Output Parameters****4.2.1 Condition Code Byte**

This byte informs you whether the SEND function was executed correctly and completely.

**Errors**

If an error occurs, you must change the program/parameters.

**Error numbers (evaluation of the condition code byte as a byte)**

- (65) The parameter "receiving CPU" is illegal. The following errors are possible:
  - The number of the receiving CPU is greater than 4.
  - The number of the receiving CPU is less than 4
  - The number of the receiving CPU is the same as the CPU's own number.

- **(67)** The special function organization block call is wrong. The following errors are possible
  - a) Secondary error, since the INITIALIZE function could not be called or was terminated by an initialization conflict.
  - b) Double call: the call for this function, SEND, SEND TEST, RECEIVE or RECEIVE TEST is illegal, since one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE or RECEIVE TEST has already been called in this CPU in a lower processing level (e.g. cyclic program execution). (See "Calling and nesting the special function organization blocks".)
  - c) The CPU's own number is incorrect (system data corrupted); following power down/power up the CPU number is generated again by the system program.
  
- **(68)** The management data (queue management) of the selected connections are incorrect; set up the buffer in the coordinator 923C again using the INITIALIZE function.
  
- **(69)** The parameter "block ID" is illegal. The following errors are possible:
  - The block ID is less than 1.
  - The block ID is greater than 2.
  
- **(70)** The parameter "block number" is illegal, since it is a data block with a special significance. The following errors are possible:
  - If the block ID = 1 : DB 0, DB 1, DB 2
  - If the block IF = 2 : DX 0, DX 1, DX 2
  
- **(71)** The parameter "block number" provided by the sender (RECEIVE) is incorrect. The specified data block does not exist.
  
- **(72)** The parameter "field number" is incorrect. The data block is too short or the field number too high.

### **Warning**

The function could be executed; the function call must be repeated, e.g. in the next cycle.

### **Warning numbers (evaluation of the condition code byte as a byte)**

- (129) The SEND function cannot transfer data, since the transmitting capacity was already zero when the function was called.

### **Initialization conflict**

The "initialization conflict" number group cannot occur with the SEND function.

## **4.2.2 Transmitting Capacity**

The "transmitting capacity" indicates how many blocks of data can still be sent and buffered.

## 5 SEND TEST Function (OB 203)

### Call parameters

#### 1. Data field:

Before calling OB 203, you must specify the input parameters in the data field. OB 203 requires 4 F flag bytes in the data field for input and output parameters:

|           |                       |                  |
|-----------|-----------------------|------------------|
| FY x + 0: | receiving CPU         | input parameter  |
| FY x + 1: | —                     | not used         |
| FY x + 2: | condition code byte   | output parameter |
| FY x + 3: | transmitting capacity | output parameter |

#### 2. ACCU-1-L:

No. of the first flag byte "x" in the data field,  
permitted values: ACCU-1-LH: 0  
ACCU-1-LL: 0 to 246

The SEND TEST function determines the number of free memory fields in the buffer of the 923C coordinator.

Depending on this number m, the SEND function can be called m times to transfer m blocks of data.

### 5.1 Input Parameters

#### 5.1.1 Receiving CPU

The CPU's own number and the number of the receiving CPU identify the connection for which the transmitting capacity (see above) is determined.

### 5.2 Output Parameters

#### 5.2.1 Condition Code Byte

This byte indicates whether the SEND TEST function was executed correctly and completely.

### Errors

If an error occurs, you must change the program parameters.

**Error numbers (evaluation of the condition code byte as a byte)**

- **(65)** The parameter "receiving CPU" is illegal. The following errors are possible:
  - The number of the receiving CPU is greater than 4.
  - The number of the receiving CPU is less than 1.
  - The number of the receiving CPU is the same as the CPU's own number.
  
- **(67)** The special function organization block call is wrong. The following errors are possible:
  - a) Secondary error, since the INITIALIZE function could not be called or was terminated by an initialization conflict.
  - b) Double call: the call for this function, SEND, SEND TEST, RECEIVE or RECEIVE TEST is illegal, since one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE or RECEIVE TEST has already been called in this CPU in a lower processing level (e.g. cyclic program execution). (See "Calling and nesting the special function organization blocks".)
  - c) The CPU's own number is incorrect (system data corrupted); following power down/power up the CPU number is generated again by the system program
  
- **(68)** The management data (queue management) of the selected connections are incorrect; set up the buffer in the coordinator 923C again using the INITIALIZE function.

**Warning**

The "warning" number group cannot occur with the SEND TEST function.

**Initialization conflict**

The "initialization conflict" number group cannot occur with the SEND TEST function.

**5.2.2 Transmitting Capacity**

The "transmitting capacity" parameter indicates how many blocks of data can be sent and buffered.



## 6 RECEIVE Function (OB 204)

### Call parameters

#### 1. Data field

Before calling OB 204, you must specify the input parameters in the data field. OB 204 requires 10 F flag bytes in the data field for input and output parameters:

|           |                      |                  |
|-----------|----------------------|------------------|
| FY x + 0: | transmitting CPU     | input parameter  |
| FY x + 1: | —                    | not used         |
| FY x + 2: | condition code byte  | output parameter |
| FY x + 3: | receiving capacity   | output parameter |
| FY x + 4: | block ID             | output parameter |
| FY x + 5: | block number         | output parameter |
| FY x + 6: | address of the first | output parameter |
| FY x + 7: | received data word   | output parameter |
| FY x + 8: | address of the last  | output parameter |
| FY x + 9: | received data word   | output parameter |

#### 2. ACCU-1-L:

No. of the first flag byte "x" in the data field,  
permitted values:

ACCU-1-LH: 0  
ACCU-1-LL: 0 to 246

The RECEIVE function takes a block of data from the buffer of the 923C coordinator. It also indicates how many data blocks are still buffered and can still be received.

The RECEIVE function should be called in a loop until all the buffered blocks of data have been received.

### 6.1 Input Parameters

#### 6.1.1 Transmitting CPU

The receive block receives data supplied by the transmitting CPU; the permitted value is between 1 and 4, but must be different from the CPU's own number.

## 6.2 Output Parameters

### 6.2.1 Condition Code Byte

This byte informs you whether the RECEIVE function was executed correctly and completely.

#### Errors

If an error occurs, you must change the program/parameters.

#### Error numbers (evaluation of the condition code byte as a byte)

- (66) The parameter "transmitting CPU" is illegal. The following errors are possible:
  - The number of the transmitting CPU is greater than 4.
  - The number of the transmitting CPU is less than 1.
  - The number of the transmitting CPU is the same as the CPU's own number.
- (67) The special function organization block call is wrong. The following errors are possible
  - a) Secondary error, since the INITIALIZE function could not be called or was terminated by an initialization conflict.
  - b) Double call: the call for this function, SEND, SEND TEST, RECEIVE or RECEIVE TEST is illegal, since one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE or RECEIVE TEST has already been called in this CPU in a lower processing level (e.g. cyclic program execution). (See "Calling and nesting the special function organization blocks".)
  - c) The CPU's own number is incorrect (system data corrupted); following power down/power up the CPU number is generated again by the system program.
- (68) The management data (queue management) of the selected connections are incorrect; set up the buffer in the coordinator 923C again using the INITIALIZE function.
- (69) The block identifiers supplied by the transmitter is illegal. The following errors are possible
  - The block ID is less than 1
  - The block ID is greater than 2.

- (70) The block number supplied by the transmitter is illegal, since it is a data block with a special significance. The following errors are possible:
  - If the block ID = 1 : DB 0, DB 1, DB 2
  - If the block ID = 2 : DX 0, DX 1, DX 2
- (71) The block number provided by the transmitter is incorrect. The specified data block does not exist.
- (73) The data block is too small to receive the block of data supplied by the transmitter.

### Warning

The function could be executed; the function call must be repeated, e.g. in the next cycle.

### Warning numbers (evaluation of the condition code byte as a byte)

- (130) The RECEIVE function cannot receive data, since the receiving capacity was already zero when the function was called.

### Initialization conflict

The "initialization conflict" number group cannot occur with the RECEIVE function.

## 6.2.2 Receiving Capacity

The "receiving capacity" parameter indicates how many blocks of data are still buffered and can still be received.

## 6.2.3 Block ID and Number

### Block ID

ID = 1: DB data block  
ID = 2: DX data block

### **Block number**

The block number along with the block ID (see above) and the addresses of the first and last data word (see below) specifies the area in which the received data were stored by the RECEIVE function (and the area from which they were taken in the transmitting CPU by the SEND function).

Remember that the receive data blocks should be in a random access memory (RAM); using read-only memories (EPROM) might possibly serve a practical purpose for transmit data blocks.

### **6.2.4 Address of the First/Last Received Data Word**

The difference between the addresses of the first and last data word transferred is a maximum of 31, since a maximum of 32 data words can be transferred per function call.

## 7 RECEIVE TEST Function (OB 205)

### Call parameters

#### 1. Data field:

Before calling OB 205, you must specify the input parameters in the data field. OB 205 requires 4 F flag bytes in the data field for input and output parameters:

|           |                       |                  |
|-----------|-----------------------|------------------|
| FY x + 0: | transmitting CPU      | input parameter  |
| FY x + 1: | —                     | not used         |
| FY x + 2: | condition code byte   | output parameter |
| FY x + 3: | transmitting capacity | output parameter |

#### 2. ACCU-1-L:

No. of the first flag byte "x" in the data field,  
permitted values: ACCU-1-LH: 0  
ACCU-1-LL: 0 to 246

The RECEIVE TEST function determines the number of occupied memory fields in the buffer of the 923C coordinator. Depending on this number m, the RECEIVE function can be called m times to receive m blocks of data.

### 7.1 Input Parameters

#### 7.1.1 Transmitting CPU

The CPU's own number and the number of the transmitting CPU identify the connection for which the receiving capacity (see above) is determined.

### 7.2 Output Parameters

#### 7.2.1 Condition Code Byte

This byte indicates whether the RECEIVE TEST function was executed correctly and completely.

### Errors

If an error occurs, you must change the program parameters.

**Error numbers (evaluation of the condition code byte as a byte)**

- (66) The parameter "transmitting CPU" is illegal. The following errors are possible:
  - The number of the transmitting CPU is greater than 4.
  - The number of the transmitting CPU is less than 1
  - The number of the transmitting CPU is the same as the CPU's own number.
- (67) The special function organization block call is wrong. The following errors are possible:
  - a) Secondary error, since the INITIALIZE function could not be called or was terminated by an initialization conflict.
  - b) Double call: the call for this function, SEND, SEND TEST, RECEIVE or RECEIVE TEST is illegal, since one of the functions INITIALIZE, SEND, SEND TEST, RECEIVE or RECEIVE TEST has already been called in this CPU in a lower processing level (e.g. cyclic program execution). (See "Calling and nesting the special function organization blocks".)
  - c) The CPU's own number is incorrect (system data corrupted); following power down/power up the CPU number is generated again by the system program..
- (68) The management data (queue management) of the selected connections are incorrect; set up the buffer in the coordinator 923C again using the INITIALIZE function.

**Warning**

The "warning" number group cannot occur with the RECEIVE TEST function.

**Initialization conflict**

The "initialization conflict" number group cannot occur with the RECEIVE TEST function.

**7.2.2 Receiving Capacity**

The "receiving capacity" parameter indicates how many blocks of data can be received and buffered.

## 8 Applications

When using one of the function blocks listed below and using interrupts (e.g. OB 2), make sure that the scratchpad flags are saved at the beginning of the interrupt handling and are written back again at the end.

### REMEMBER



This also applies to the setting "interrupts at block boundaries", since the call of the special function organization blocks represents a block boundary.

### 8.1 Calling the Special Function OB Using Function Blocks

The following five function blocks (FB 200 and FB 202 to FB 205) contain the call for the corresponding special function organization block for multiprocessor communication (OB 200 and OB 202 to OB 205).

The numbers of the function blocks are not fixed and can be changed. The parameters of the special function OBs are transferred as actual parameters when the function blocks are called. The direct call of the special function organization blocks is faster, however, is more difficult to read owing to the absence of formal parameters.

| FB no. | FB name  | Function                  |
|--------|----------|---------------------------|
| FB 200 | INITIAL  | Set up buffer             |
| FB 202 | SEND     | Send a block of data      |
| FB 203 | SEND-TST | Test the sending capacity |
| FB 204 | RECEIVE  | Receive a block of data   |
| FB 205 | RECV-TST | Test receiving capacity   |

The flag area from FY 246 to maximum FY 255 is used by the function blocks as a parameter field for the special function organization blocks.

The exact significance of the input and output parameters is explained in the description of the special function organization blocks.

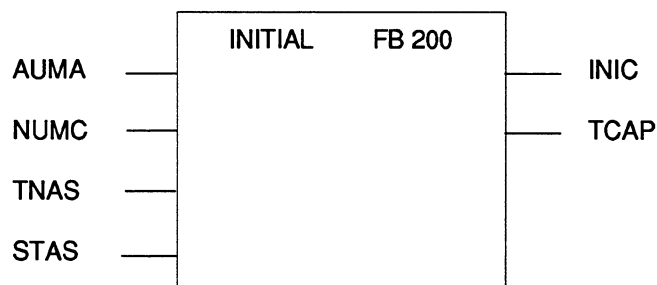
### REMEMBER



The following examples of applications involve finished applications that you can program by copying them.

### 8.1.1 Setting Up a Buffer (FB 200)

| Parameter name | Significance   | Parameter type | Data type | Parameter field |
|----------------|--|----------------|-----------|-----------------|
| AUMA           | Automatic/manual   | I              | BY        | FY 246          |
| NUMC           | Number of CPUs   | I              | BY        | FY 247          |
| TNAS           | Type (H byte) and number (L byte) of the data block containing the assignment list | I              | W         | FW 248          |
| STAS           | Start address of the assignment list   | I              | W         | FW 250          |
| INIC           | Initialization conflict  | Q              | BY        | FY 252          |
| TCAP           | Total capacity   | Q              | BY        | FY 253          |



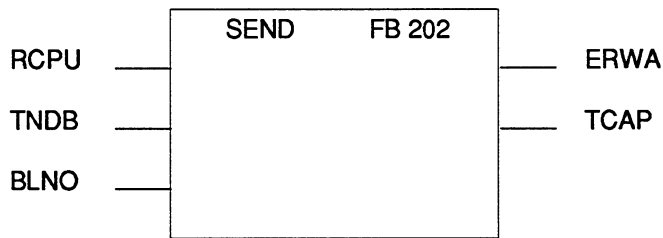


|              |              |   |            |     |
|--------------|--------------|---|------------|-----|
| FB 200       |              |   | LEN=45     | ABS |
| SEGMENT 1    |              |   |            |     |
| NAME:INITIAL |              |   |            |     |
| DECL :AUMA   | I/Q/D/B/T/C: | I | BI/BY/W/D: | BY  |
| DECL :NUMC   | I/Q/D/B/T/C: | I | BI/BY/W/D: | BY  |
| DECL :TNAS   | I/Q/D/B/T/C: | I | BI/BY/W/D: | W   |
| DECL :STAS   | I/Q/D/B/T/C: | I | BI/BY/W/D: | W   |
| DECL :INIC   | I/Q/D/B/T/C: | Q | BI/BY/W/D: | BY  |
| DECL :TCAP   | I/Q/D/B/T/C: | Q | BI/BY/W/D: | BY  |

|       |     |        |   |
|-------|-----|--------|---|
| 0017  | :L  | =AUMA  | AUTOMATIC/MANUAL                        |
| 0018  | :T  | FY 246 |   |
| 0019  | :L  | =NUMC  | NUMBER OF CPUs                          |
| 001A  | :T  | FY 247 |   |
| 001B  | :L  | =TNAS  | DB TYPE, DB NO.                         |
| 001C  | :T  | FW 248 |   |
| 001D  | :L  | =STAS  | START ADDRESS OF THE<br>ASSIGNMENT LIST |
| 001E  | :T  | FW 250 |   |
| 001 F | :   |        |   |
| 0020  | :L  | KB 246 | SF OB:                                  |
| 0021  | :JU | OB 200 | INITIALIZE                              |
| 0022  | :   |        |   |
| 0023  | :L  | FY 252 | INITIALIZATION CONFLICT                 |
| 0024  | :T  | =INIC  |   |
| 0025  | :L  | FY 253 | TOTAL CAPACITY                          |
| 0026  | :T  | =TCAP  |   |
| 0027  | :BE |        |   |

### 8.1.2 Sending a Block of Data (FB 202)

| Parameter name | Significance   | Parameter type | Data type | Parameter field |
|----------------|--|----------------|-----------|-----------------|
| RCPU           | Receiving CPU  | I              | BY        | FY 246          |
| TNDB           | Type (H byte) and number (L byte) of the source data block | I              | W         | FW 247          |
| BLNO           | Block number   | I              | BY        | FY 249          |
| ERWA           | Error warning  | Q              | BY        | FY 250          |
| TCAP           | Transmitting capacity                                      | Q              | BY        | FY 251          |

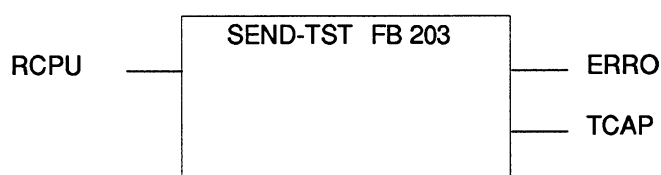


|            |              |   |            |     |
|------------|--------------|---|------------|-----|
| FB 202     |              |   | LEN=40     | ABS |
| SEGMENT 1  |              |   |            |     |
| NAME:SEND  |              |   |            |     |
| DECL :RCPU | I/Q/D/B/T/C: | I | BI/BY/W/D: | BY  |
| DECL :TNDB | I/Q/D/B/T/C: | I | BI/BY/W/D: | W   |
| DECL :BLNO | I/Q/D/B/T/C: | I | BI/BY/W/D: | BY  |
| DECL :ERWA | I/Q/D/B/T/C: | Q | BI/BY/W/D: | BY  |
| DECL :TCAP | I/Q/D/B/T/C: | Q | BI/BY/W/D: | BY  |

|      |     |               |                       |
|------|-----|---------------|-----------------------|
| 0014 | :L  | =RCPU         | RECEIVING CPU         |
| 0015 | :T  | FY 246        |                       |
| 0016 | :L  | =TNDB         | DB TYPE, DB NO.       |
| 0017 | :T  | FW 247        |                       |
| 0018 | :L  | =BLNO         | BLOCK NUMBER          |
| 0019 | :T  | FY 249        |                       |
| 001A | :   |               |                       |
| 001B | :L  | KB 246        | SF OB:                |
| 001C | :JU | <b>OB 202</b> | SEND A BLOCK OF DATA  |
| 001D | :   |               |                       |
| 001E | :L  | FY 250        | ERROR/WARNING         |
| 001F | :T  | =ERWA         |                       |
| 0020 | :L  | FY 251        | TRANSMITTING CAPACITY |
| 0021 | :T  | =TCAP         |                       |
| 0022 | :BE |               |                       |

## 8.1.3 Testing the Transmitting Capacity (FB 203)

| Parameter name | Significance          | Parameter type | Data type | Parameter field |
|----------------|-----------------------|----------------|-----------|-----------------|
| RCPU           | Receiving CPU         | I              | BY        | FY 246          |
| ERRO           | Error                 | Q              | BY        | FY 248          |
| TCAP           | Transmitting capacity | Q              | BY        | FY 249          |



```

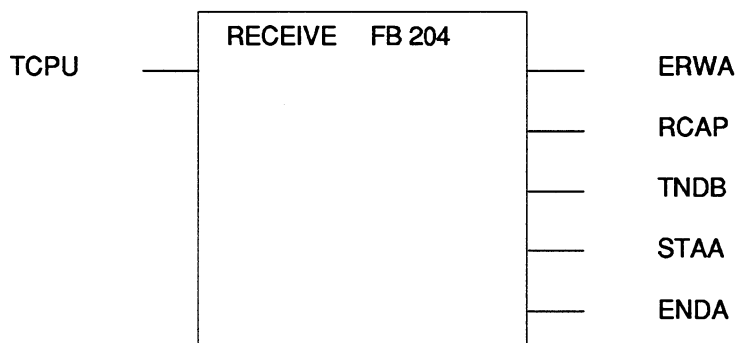
FB 203                               LEN=30  ABS
SEGMENT 1
NAME:SEND-TST
DECL :RCPU      I/Q/D/B/T/C:      I      BI/BY/W/D:      BY
DECL :ERRO      I/Q/D/B/T/C:      I      BI/BY/W/D:      BY
DECL :TCAP      I/Q/D/B/T/C:      Q      BI/BY/W/D:      BY
  
```

```

000E      :L      =RCPU              RECEIVING CPU
000F      :T      FY 246
0010      :
0011      :L      KB 246              SF OB:
0012      :JU      OB 203            TEST TRANSMITTING CAPACITY
0013      :
0014      :L      FY 248              ERROR
0015      :T      =ERRO
0016      :L      FY 249              TRANSMITTING CAPACITY
0017      :T      =TCAP
0018      :BE
  
```

### 8.1.4 Receiving a Block of Data (FB 204)

| Parameter name | Significance  | Parameter type | Data type | Parameter field |
|----------------|---|----------------|-----------|-----------------|
| TCPU           | Transmitting CPU  | I              | BY        | FY 246          |
| ERWA           | Error warning   | Q              | BY        | FY 248          |
| RCAP           | Receiving capacity  | Q              | BY        | FY 249          |
| TNDB           | Type (H byte) and number (L byte) of the destination data block | Q              | W         | FW 250          |
| STAA           | Address of the first received data word (start address)         | Q              | W         | FW 252          |
| ENDA           | Address of the last received data word (end address)            | Q              | W         | FW 254          |



```

FB 204                                LEN=45  ABS
SEGMENT 1
NAME:RECEIVE
DECL :TCPU      I/Q/D/B/T/C:      I  BI/BY/W/D:      BY
DECL :ERWA      I/Q/D/B/T/C:      Q  BI/BY/W/D:      BY
DECL :RCAP      I/Q/D/B/T/C:      Q  BI/BY/W/D:      BY
DECL :TNDB      I/Q/D/B/T/C:      Q  BI/BY/W/D:      W
DECL :STAA      I/Q/D/B/T/C:      Q  BI/BY/W/D:      W
DECL :ENDA      I/Q/D/B/T/C:      Q  BI/BY/W/D:      W

```

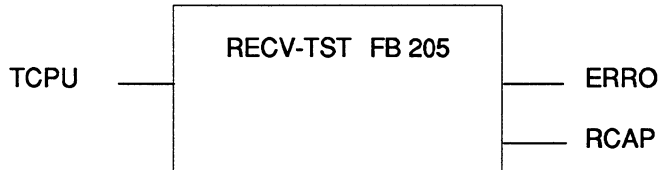
```

0017      :L      =TCPU              TRANSMITTING CPU
0018      :T      FY 246
0019      :
001A      :L      KB 246              SF OB:
001B      :JU     OB 204              RECEIVE A BLOCK
                                      OF DATA
001C      :
001D      :L      FY 248              ERROR/WARNING
001E      :T      =ERWA
001F      :L      FY 249              RECEIVING CAPACITY
0020      :T      =RCAP
0021      :L      FW 250              DB TYPE, DB NO.
0022      :T      =TNDB
0023      :L      FW 252              START ADDRESS
0024      :T      =STAA
0025      :L      FW 254              END ADDRESS
0026      :T      =ENDA
0027      :BE

```

### 8.1.5 Testing the Receiving Capacity (FB 205)

| Parameter name | Significance       | Parameter type | Data type | Parameter field |
|----------------|--------------------|----------------|-----------|-----------------|
| TCPU           | Transmitting CPU   | I              | BY        | FY 246          |
| ERRO           | Error              | Q              | BY        | FY 248          |
| RCAP           | Receiving capacity | Q              | BY        | FY 249          |



```

FB 205                LEN=30  ABS
SEGMENT 1
NAME: RECV-TST
DECL :TCPU           I/Q/D/B/T/C:  I  BI/BY/W/D:  BY
DECL :ERRO           I/Q/D/B/T/C:  Q  BI/BY/W/D:  BY
DECL :RCAP           I/Q/D/B/T/C:  Q  BI/BY/W/D:  BY
  
```

```

000E   :L   =TCPU                TRANSMITTING CPU
000F   :T   FY 246
0010   :
0011   :L   KB 246                SF OB:
0012   :JU  OB 205                TEST RECEIVING CAPACITY
0013   :
0014   :L   FY 248                ERROR
0015   :T   =ERRO
0016   :L   FY 249                RECEIVING CAPACITY
0017   :T   =RCAP
0018   :BE
  
```

## 8.2 Transferring Data Blocks

### 8.2.1 Functional Description

The function block TRAN DAT (FB 110) transfers a selectable number of blocks of data from a data block in one CPU to the data block of the same type and same number in a different CPU. (For a description of the parameter list, function block and STEP 5 program, see the following page.)

The FB number has been selected at random and you can use other numbers.

### 8.2.2 Transferring a Data Block (FB 110)

The data area to be transferred is stipulated by the input parameter FIRB (= number of the first block of data to be transferred) and NUMB (= number of blocks of data to be transferred). A block of data normally consists of 32 data words. Depending on the data block length, the last block of data may be less than 32 data words.

The transfer is triggered by a positive-going edge at the start input STAR. If the output parameter REST is zero after the transfer, this means that the function block TRANDAT was able to send all the blocks of data (according to the NUMB parameter).

If, however, the REST output parameter has a value greater than zero, this means that the function block must be called again, for example in the next cycle. This means that you or the user program can only change the set of parameters (i.e. the values of all parameters) when the REST parameter indicates zero showing that the data transfer is complete.

You can call the function block TRANDAT several times with different parameters. In this case, various data areas are transferred simultaneously (interleaved in each other). The special function organization blocks for multiprocessor communication OB 202 to OB 205 can also be used "directly". This possibility is illustrated in the application example.

If the SEND function (OB 202) is not correctly executed within the TRANDAT function block, the error number is entered in the output parameter ERRO, the RLO = "1" and the output parameter REST is set to "0".

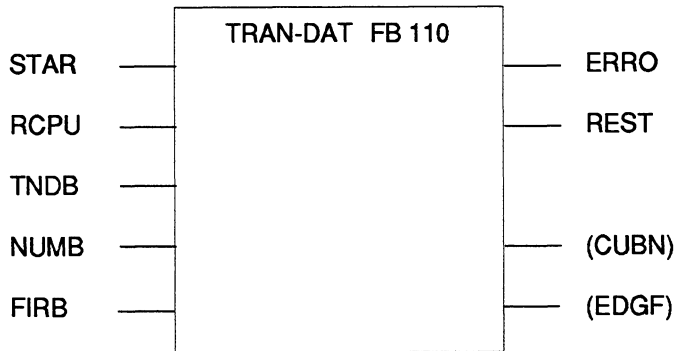
The TRANDAT function block uses flag bytes FY 246 to FY 251 as scratchpad flags. All other variables whose value is significant as long as the output parameter REST = "0" continue to have memory assigned to them using the mechanism of formal/actual parameters. This is necessary to allow various data blocks to be transferred simultaneously.

Note: data block and block of data are not synonymous. Data block is a DB or DX and a block of data is part of a DB from 1 to maximum 32 data words.



| Parameter name     | Significance  | Parameter type | Data type |
|--------------------|---|----------------|-----------|
| STAR               | Start the transfer of the data block on a positive-going edge         | I              | BI        |
| RCPU               |   | I              | BY        |
| TNDB               | Receiving CPU   | I              | W         |
| NUMB               | Type (H byte) and number (L byte) of the data block to be transferred | I              | BY        |
| FIRB               | Number of blocks of data to be transferred                            | I              | BY        |
| ERRO               | Number of the first block of data to be transferred                   | Q              | BY        |
| REST               | Error   | Q              | BY        |
| CUBN <sup>1)</sup> | Number of blocks of data still to be transferred                      | Q              | BY        |
| EDGF <sup>1)</sup> | Current block number  | Q              | BI        |
|                    | Edge flag   |                |           |

1) Internal scratchpad flag, not intended for evaluation.



```

FB 110                                LEN=89   ABS
SEGMENT 1
NAME:TRAN-DAT
DECL :STAR      I/Q/D/B/T/C:  I   BI/BY/W/D:  BI
DECL :RCPU      I/Q/D/B/T/C:  I   BI/BY/W/D:  BY
DECL :TNDB      I/Q/D/B/T/C:  I   BI/BY/W/D:  W
DECL :NUMB      I/Q/D/B/T/C:  I   BI/BY/W/D:  BY
DECL :FIRB      I/Q/D/B/T/C:  I   BI/BY/W/D:  BY
DECL :ERRO      I/Q/D/B/T/C:  Q   BI/BY/W/D:  BY
DECL :REST      I/Q/D/B/T/C:  Q   BI/BY/W/D:  BY
DECL :CUBN      I/Q/D/B/T/C:  Q   BI/BY/W/D:  BY
DECL :EDGF      I/Q/D/B/T/C:  Q   BI/BY/W/D:  BI

0020      :L      =RCPU                ASSIGN PARAMETER FIELD FOR
0021      :T      FY 246                SF OB 202
0022      :L      =TNDB
0023      :T      FW 247
0024      :
0025      :L      =REST                FIRST SEND ANY REMAINING
0026      :L      KB 0                BLOCKS OF DATA
0027      :><F
0028      :JC     =TRAN
0029      :
002A      :AN     =STAR                POSITIVE EDGE AT START
002B      :RB     =EDGF                INPUT ?
002C      :ON     =STAR
002D      :O      =EDGF
002E      :JC     =GOOD
002F      :S      =EDGF
0030      :
0031      :L      =NUMB                INITIALIZE THE GLOBAL FLAGS
0032      :T      =REST                AFTER POSITIVE EDGE AT
0033      :L      =FIRB                START INPUT
0034      :T      =CUBN
0035      :
0036      :L      =REST                AS LONG AS REST ><0,
0038 LOOP :L      KF+0                CONTINUE TO ATTEMPT
0039      :!=F                        TO SEND BLOCKS OF DATA
003A      :JC     =GOOD
003B TRAN :L      =CUBN
003C      :T      FY 249
003D      :L      KB 246                SF OB:
003E      :JU     OB 202            SEND A BLOCK OF DATA
003F      :L      FY 250
0040      :JM     =ERRO                ABORT IF ERROR
0041      :JP     =GOOD                ABORT IF TRANS-CAP = 0
0042      :L      =CUBN                INCREMENT BLOCK NUMBER
0043      :I      1
0044      :T      =CUBN
0045      :L      =REST                DECREMENT NUMBER OF
0046      :D      1                REMAINING BLOCKS OF DATA

```

```

0047      :T  =REST
0048      :JU =LOOP
0049      :
004A GOOD:A  F 0.0          REGULAR END OF PROGRAM
004B      :AN  F 0.0
004C      :L   KB 0          RLO = 0, ERRO = 0
004D      :T  =ERRO
004E      :BEU
004F      :
0050 ERRO :T  =ERRO          PROGRAM END IF ERROR
0051      :L   KB 0
0052      :T  =REST          RLO = 1, ERROR CONTAINS
                                ERROR NUMBER
0053      :BE

```

### 8.2.3 Application Example (for the S5-135U)

You want CPU 1 to transfer data blocks DB 3 (blocks of data 2 to 5) and DB 4 (blocks of data 1 to 3) to CPU 2 during the cyclic user program. The RECEIVE function (OB 204) is also called in the cyclic user program.

The following blocks must be loaded in the individual CPUs:

| Function                  | CPU 1      | CPU 2      |
|---------------------------|------------|------------|
| Cold restart block        | OB 20      | —          |
| Cycle block <sup>1)</sup> | FB 0       | FB 0       |
| Send DB                   | DB 3; DB 4 | —          |
| Receive DB                | —          | DB 3; DB 4 |

<sup>1)</sup> Only OB 1 is permitted as the cycle block in the CPU 946/947.

OB 20 calls the INITIALIZE function (OB 200) and reserves several memory fields for the connection from CPU 1 to CPU 2.

The cyclic user program in function block FB 0 of CPU 1 contains two calls for the function block TRANDAT in each case with different sets of parameters. The transfer of the first data block DB 3 begins after a positive edge after input I 2.0. A positive edge at input I 2.1 starts the transfer of the second data block DB 4.

```

FB 0                               LEN=66   ABS
SEGMENT 1
NAME:DEMO

0005      :L   KB 2                TO CPU 2 ..
0006      :T   FY 0
0007      :L   KY 1,3              .. FROM DATA BLOCK DB 3
0009      :T   FW 1
000A      :L   KB 4                .. FOUR BLOCKS OF DATA
000B      :T   FY 3
000C      :L   KB 2                .. SEND FROM 2ND BLOCK OF DATA
000D      :T   FY 4
000E      :
000F      :JU  FB 110
0010 NAME :TRAN-DAT
0011 STAR :   I 2.0
0012 RCPU :   FY 0
0013 TNDB :   FW 1
0014 NUMB :   FY 3
0015 FIRB :   FY 4
0016 ERRO :   FY 5
0017 REST :   FY 6
0018 CUBN :   FY 7
0019 EDGF :   F 8.0
001A      :
001B      :
001C      :JC  =HALT              ABORT AFTER ERROR
001D      :
001E      :L   KB 2                TO CPU 2 ..
001F      :T   FY 10
0020      :L   KY 1,4              .. FROM DATA BLOCK DB 4
0022      :T   FW 11
0023      :L   KB 3                .. THREE BLOCKS OF DATA
0024      :T   FY 13
0025      :L   KB 1                .. SEND FROM 1ST BLOCK OF DATA
0026      :T   FY 14
0027      :
0028      :JU  FB 110
0029 NAME :TRAN-DAT
002A STAR :   I 2.1
002B RCPU :   FY 10
002C TNDB :   FW 11
002D NUMB :   FY 13
002E FIRB :   FY 14
002F ERRO :   FY 5
0030 REST :   FY16
0031 CUBN :   FY17
0032 EDGF :   F 8.1
0033      :
0034      :
0035      :JC  =HALT              ABORT AFTER ERROR
0036      :BEU

```

0037 :  
0038 HALT :

The error handling takes place here (e.g stop, message output on the printer, ...)

0039 :BE

In CPU 2, the RECEIVE function (OB 204) called by FB 0 enters each transmitted block of data into the appropriate data block. It may take several cycles before a data block has been completely received.

FB 0  
SEGMENT 1  
NAME:RECV-DAT

LEN=26 ABS

|           |      |        |                                |
|-----------|------|--------|--------------------------------|
| 0005      | :L   | KB 1   | RECEIVE DATA FROM CPU 1        |
| 0006      | :T   | FY 246 |                                |
| 0007      | :    |        |                                |
| 0008 LOOP | :L   | KB 246 | SF OB:                         |
| 0009      | :JU  | OB 204 | RECEIVE                        |
| 000A      | :JM  | =ERRO  | ABORT IF ERROR                 |
| 000B      | :L   | FY 249 | THE RECEIVE FUNCTION IS        |
| 000C      | :L   | KB 0   | CALLED UNTIL THERE ARE NO      |
| 000D      | :><F |        | FURTHER BLOCKS OF DATA IN      |
| 000E      | :JC  | =LOOP  | THE BUFFER, I.E. THE RECEIVING |
| 000F      | :    |        | CAPACITY = 0.                  |
| 0010      | :BEU |        |                                |
| 0011 ERRO | :    |        |                                |

The error handling takes place here (e.g. stop, message output on printer, ...)

0012 :BE

## 8.3 Extending the IPC Flag Area

### 8.3.1 The Problem

In the multiprocessor programmable controllers S5-135U and S5-155U, each of the 256 flag bytes of a CPU can become an input or output IPC flag by making an entry in data block DB 1. This, however, reduces the number of "normal" flag bytes. To transfer a data record (several bytes) other mechanisms are also required (semaphore variable or DX 0 parameter assignment "transfer IPC flags as a block") are necessary to prevent the receiver from receiving a fragmented data record.

### 8.3.2 The Solution

Consecutive data words of a DB or DX data block are defined from DW 0 onwards as "IPC data words". Each connection is assigned its own data block and is totally independent of the other connections.

At the beginning of the cycle block (CPU 946/947: OB 1, CPU 92x: OB 1 or FB 0), the IPC data words are received with the aid of the special function organization blocks for multiprocessor communication. This is followed by the "regular" cyclic program, that evaluates the received data and generates the data to be sent. At the end of the cycle, this data is then sent with the aid of the special organization blocks for multiprocessor communication. It can therefore be received by the other CPUs at the beginning of their cycles.

The following applies for each of the maximum 12 possible connections regardless of the other connections:

- The transmitting CPU is only active when the receiving CPU has read out all the "old" data from the 923C buffer.
- The receiving CPU is only active when the transmitting CPU has written all the "new" data in the 923C buffer.

This means that the receiving CPU can either receive a complete new data record or the old data record remains unchanged: **no mixing of "old" and "new" data.**

### 8.3.3 Data Structure

Which data words (for the data word area below) are to be transferred from which CPU to which CPU is described in the connection list (see table on the following page). This is located in an additional data block that must exist in all the CPUs involved.

The data word areas always begin from data word DW 0, and their lengths are specified in blocks of data. Remember the following points:

- A complete block of data consists of 32 data words.
- If the last block of a data block is "truncated", i.e. it contains between 1 and 31 data words, less data words are transferred.
- If a send data block is longer than the number of blocks of data specified in the connection list, the excess data words can be used in the corresponding CPU.
- If a receive data block is longer than the received data word area, the excess data words can be used in the corresponding CPU.



The connection consists of two similarly structured sub-lists, each with 16 data words. For each of the four sender CPUs (S1, S2, S3, S4) three entries are required to describe a connection.

- **Number of blocks of data**

The number of blocks of data specifies the size (= the number of data words) of the data word area to be transferred. (If connections do not exist or you do not require them, enter 0 for the number of blocks of data, and for the DB type and DB number.)

- **DB type**

Type of data block containing the data word area to be transferred.

- **DB number**

Number of the data block containing the data word area to be transferred.

As shown in the table, these entries can be read in and completed in lines. If, for example, you want to transfer the first two blocks of data in data block DB 10 from CPU 2 (S2) to CPU 3, make the following entries:

CPU 2 (**S 2**) sends ..

|       |       |                                  |  |      |    |     |
|-------|-------|----------------------------------|--|------|----|-----|
| DW 22 | 3     | 2                                |  | DW 6 | 1  | 10  |
|       | ↓     | ↓                                |  | ↓    | ↓  | ↓   |
| .. to | CPU 3 | 2 blocks of data from data block |  |      | DB | 10. |

Sub-list 2 is identical to the assignment ("manual" mode) required for the INITIALIZE function (OB 200). Within the data block, sub-list 1 must occupy data words 0 to 15 and sub-list 2 data words 16 to 31. You must not alter the entries shown in bold face.



### 8.3.4 Program Structure

During restart, one of the CPUs calls the INITIALIZE function (OB 200) to reserve exactly the same number of coordinator memory fields per connection as blocks of data to be transmitted on this connection.

To send and receive data word areas, each CPU uses two function blocks:

| FB no. | Name     | Function                                    |
|--------|----------|---|
| FB 100 | SEND-DAT | Send data word areas to the other CPUs      |
| FB 101 | RECV-DAT | Receive data word areas from the other CPUs |

These FB numbers have been selected at random and you can use others.

The function blocks SEND-DAT and RECV-DAT read the connection list to determine which data word areas are to be sent from or received by which data blocks. The **whole** data word area is always sent or received. If this is not possible owing to insufficient transmitting or receiving capacity, the send or receive function is not executed.

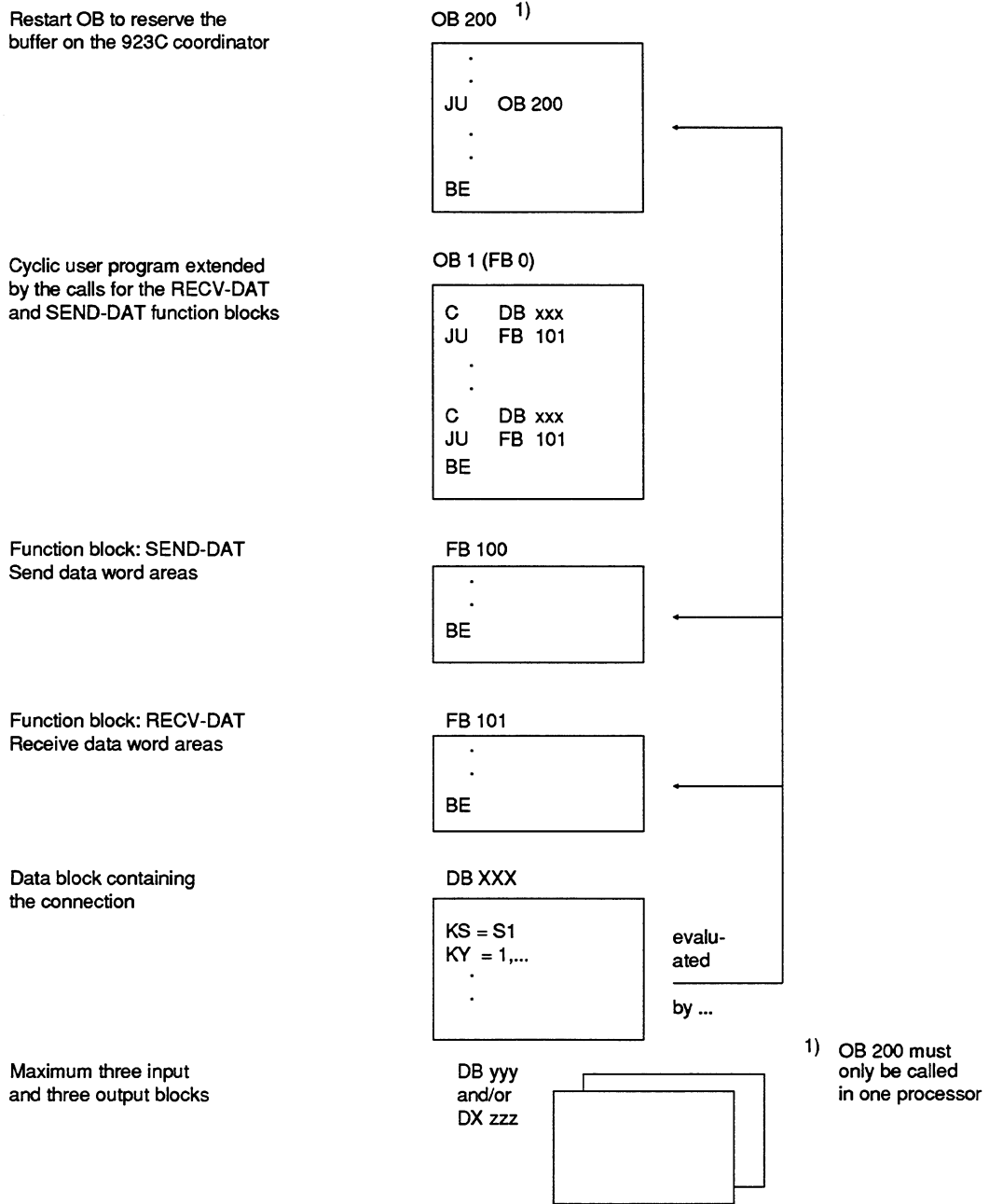


Fig. 3 Overview of the blocks required in each CPU

**REMEMBER**

The function blocks SEND-DAT and RECV-DAT contain the special function organization blocks for multiprocessor communication OB 202 to OB 205. You cannot call these organization blocks outside SEND-DAT / RECV-DAT.

### 8.3.5 Sending Data Word Areas (FB 100)

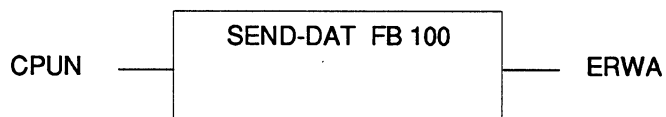
Before you call FB 100, the data block containing the connection list must be open. The function block SEND-DAT requires the number of the CPU on which it is called in order to evaluate the information contained in the connection list.

If the SEND function (OB 202) is not executed correctly in the function block, the error or warning number is transferred to the output parameter ERWA and RLO is set to 1.

If the input parameter CPUN (CPU number) is illegal, ERWA has the value 16 (bit  $2^4 = 1$ ).

The function block SEND-DAT uses flag bytes FY 239 to FY 251 as scratchpad flags.

| Parameter name | Significance   | Parameter type | Data type |
|----------------|--|----------------|-----------|
| CPUN           | Number of the CPU on which FB 100 is called. The numbers 1 to 4 are permitted. | D              | KF        |
| ERWA           | Error/warning (see SEND function (OB 202))                                     | Q              | BY        |



```

FB 100                                LEN=90
SEGMENT 1                            0000
NAME:SEND-DAT
DECL :CPUN      I/Q/D/B/T/C:      D   KM/KH/KY/KS/KF/KT/KC/KG:  KF
DECL :ERWA      I/Q/D/B/T/C:      Q   BI/BY/W/D:      BY

000B      :LW  =CPUN                CPUN = CPUN -1
000C      :L   KB 1                ERROR IF:
000D      :-F
000E      :JM  =ERWA                CPU NO. <1
000F      :L   KB 3
0010      :>F
0011      :JC  =ERWA                CPU NO. >4
0012      :TAK
0013      :
0014      :SLW      2                CPUN = CPUN * 4
0015      :T   FY 245                BASE ADDRESS
0016      :
0017      :L   KB 1
0018      :T   FY 244                CONNECTION COUNTER
0019      :
001A LOOP :L   FY 245                BASE ADDRESS
001B      :L   FY 244                + COUNTER
001C      :+F
001D      :T   FW 240
001E      :ADD BN+16                + OFFSET
001F      :T   FW 242
0020      :
0021      :DO  FW 242
0022      :L   DR 0                NUMBER OF RESERVED
0023      :T   FY 239                FIELDS = 0 ?
0024      :L   KB 0
0025      :!=F
0026      :JC  =EMPT
0027      :
0028      :DO  FW 242
0029      :L   DL 0                NO. OF THE RECEIVING CPU
002A      :T   FY 246
002B      :L   KB 246
002C      :JU  OB 203                SF OB:
002D      :L   FY 248                TEST TRANSMITTING CAPACITY
002E      :JC  =OBER                ABORT IF ERROR
002F      :
0030      :L   FY 249                TRANSMITTING CAPACITY >> NO.
0031      :L   FY 239                OF RESERVED FIELDS ?
0032      :><F
0033      :JC  =EMPT
0034      :
0035      :L   KB 0                FIELD COUNTER
0036      :T   FY 249
0037      :
0038      :DO  FW 240
0039      :L   DW 0                TYPE AND NUMBER OF THE
003A      :T   FW 247                SOURCE DB

```

|      |      |      |        |
|------|------|------|--------|
| 003B | :    |      |        |
| 003C | TRAN | :L   | KB 246 |
| 003D |      | :JU  | OB 202 |
| 003E |      | :L   | FY 250 |
| 003F |      | :JC  | =OBER  |
| 0040 | :    |      |        |
| 0041 |      | :L   | FY 249 |
| 0042 |      | :I   | 1      |
| 0043 |      | :T   | FY 249 |
| 0044 |      | :L   | FY 239 |
| 0045 |      | :<F  |        |
| 0046 |      | :JC  | =TRAN  |
| 0047 | :    |      |        |
| 0048 | EMPT | :L   | FY 244 |
| 0049 |      | :I   | 1      |
| 004A |      | :T   | FY 244 |
| 004B |      | :L   | KB 4   |
| 004C |      | :<F  |        |
| 004D |      | :JM  | =LOOP  |
| 004E |      | :L   | KB 0   |
| 004F |      | :T   | =ERWA  |
| 0050 |      | :BEU |        |
| 0051 | :    |      |        |
| 0052 | ERWA | :L   | KB 16  |
| 0053 | OBER | :T   | =ERWA  |
| 0054 |      | :BE  |        |

|                                  |
|----------------------------------|
| SF OB:                           |
| SEND A BLOCK OF DATA             |
| ABORT IF ERROR/WARNING           |
|                                  |
| BLOCK NO. = BLOCK NO. + 1        |
|                                  |
| ALL BLOCKS OF DATA TRANSFERRED ? |
|                                  |
| INCREMENT                        |
| CONNECTION COUNTER               |
|                                  |
| ALL THREE CONNECTIONS            |
| PROCESSED ?                      |
|                                  |
| REGULAR PROGRAM END:             |
| RLO = 0, ERWA = 0                |
|                                  |
| PROGRAM END IF ERROR:            |
| RLO = 1, ERWA CONTAINS           |
| ERROR/WARNING NUMBER             |

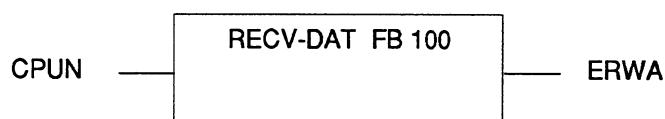
### 8.3.6 Receive Data Word Areas (FB 101)

Before you call FB 101, the data block containing the connection list must already be open. The function block RECV-DAT requires the number of the CPU in which it is called in order to evaluate the information contained in the connection list.

If the RECEIVE function (OB 204) is not correctly processed within the function block, the corresponding error or warning number is transferred to the output parameter ERWA and the RLO is set to 1. If the input parameter CPUN is illegal, ERWA has the value 16 (bit  $2^4 = 1$ ).

The RECV-DAT function block uses flag bytes FY 242 to FY 255 as scratchpad flags.

| Parameter name | Significance  | Parameter type | Data type |
|----------------|---|----------------|-----------|
| CPUN           | Number of the <b>CPU</b> on which FB 101 is called. The numbers 1 to 4 are permitted. | D              | KF        |
| ERWA           | Error/warning (see RECEIVE function (OB 204)).  | Q              | BY        |



```

FB 101                                LEN=88
SEGMENT 1                             0000
NAME:RECV-DAT
DECL :CPUN      I/Q/D/B/T/C:          D   KM/KH/KY/KS/KF/KT/KC/KG:  KF
DECL :ERWA      I/Q/D/B/T/C:          Q   BI/BY/W/D:          BY
    
```

```

000B      :LW  =CPUN                    ERROR IF:
000C      :L   KB 1
000D      :<F
000E      :JC  =ERWA                    CPU NO.<1
000F      :LW  =CPUN
0010      :L   KB 4
0011      :>F
0012      :JC  =ERWA                    CPU NO.>4
0013      :
0014      :L   KB 1                    CONNECTION COUNTER
0015      :T   FY 242
0016      :
0017      :L   KB 16
0018      :T   FW 244                    POINTER TO SUB-LIST 2
0019      :
001A SRCH :L   FW 244                    SEARCH SUB-LIST 2 UNTIL THE
001B      :I   1                        NEXT ENTRY FOR THE RECEIVING
001C      :T   FW 244                    CPU WITH THE NUMBER "CPUN"
001D      :DO  FW 244                    IS FOUND
001E      :L   DL 0
001F      :LW  =CPUN
0020      :><F
0021      :JC  =SRCH
0022      :
0023      :DO  FW 244
0024      :L   DR 0                    NUMBER OF RESERVED
0025      :T   FY 243                    MEMORY FIELDS = 0 ?
0026      :L   KB 0
0027      :!=F
0028      :JC  =EMPT
0029      :
002A      :L   FW 244
002B      :L   KM 00000000 00001100    DETERMINE THE NUMBER OF THE
002D      :AW                                     TRANSMITTING CPU FROM THE
002E      :SRW 2                                     POINTER TO SUB-LIST 2
002F      :I   1
0030      :T   FY 246
0031      :
0032      :L   KB 246                    SF OB:
0033      :JU  OB 205                    TEST RECEIVING CAPACITY
0034      :L   FY 248
0035      :JC  = OBER                    ABORT IF ERROR
0036      :
0037      :L   FY 249                    RECEIVING CAPACITY = NUMBER
0038      :L   FY 243                    OF RESERVED MEMORY FIELDS?
0039      :><
003A      :JC  =EMPT
003B      :
    
```

|      |      |      |               |                            |
|------|------|------|---------------|----------------------------|
| 003C | RECV | :L   | KB 246        | SF OB:                     |
| 003D |      | :JU  | <b>OB 204</b> | RECEIVE A BLOCK OF         |
| 003E |      | :L   | FY 248        | DATA                       |
| 003F |      | :JM  | =OBFE         | ABORT IF ERROR/            |
| 0040 |      | :    |               | WARNING                    |
| 0041 |      | :L   | FY 249        | IF RECEIVING CAPACITY = 0, |
| 0042 |      | :L   | KB 0          | PROCESS NEXT               |
| 0043 |      | :><F |               | CONNECTION                 |
| 0044 |      | :JC  | =RECV         |                            |
| 0045 |      | :    |               |                            |
| 0046 | EMPT | :L   | FY 242        | INCREMENT                  |
| 0047 |      | :I   | 1             | CONNECTION COUNTER         |
| 0048 |      | :T   | FY 242        |                            |
| 0049 |      | :L   | KB 4          | ALL CONNECTIONS            |
| 004A |      | :<F  |               | PROCESSED ?                |
| 004B |      | :JM  | =SRCH         |                            |
| 004C |      | :L   | KB 0          | REGULAR PROGRAM END:       |
| 004D |      | :T   | =ERWA         | RLO = 0, ERWA = 0          |
| 004E |      | :BEU |               |                            |
| 004F |      | :    |               |                            |
| 0050 | ERWA | :L   | KB 16         | PROGRAM END IF ERROR:      |
| 0051 | OBER | :T   | =ERWA         | RLO = 1, ERWA CONTAINS     |
| 0052 |      | :BE  |               | ERROR/WARNING NUMBER       |



### 8.3.7 Application Example (for S5-135U)

You want to exchange data between three CPUs:

- From CPU 1 to CPU 2: data block DB 3, DW 0 to DW 127 (= 4 blocks of data)
- From CPU 1 to CPU 3: data block DX 4, DW 0 to DW 63 (= 2 blocks of data)
- From CPU 2 to CPU 1 and CPU 3: data block DB 5, DW 0 to DW 95 (= 3 blocks of data)

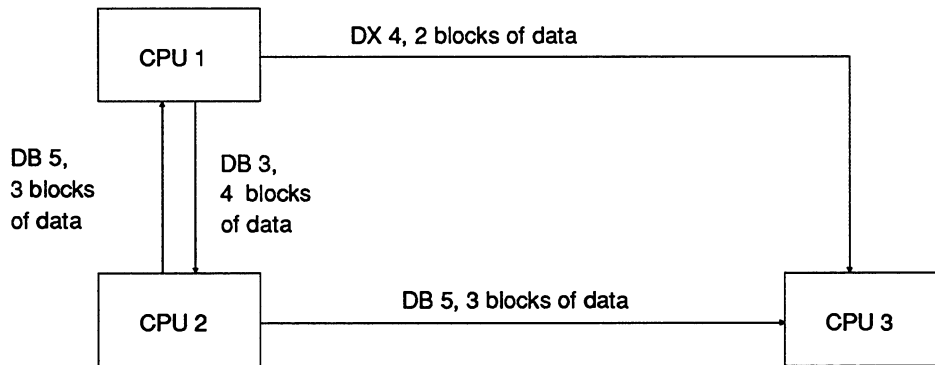


Fig. 4 Data exchange between 3 CPUs

Function block FB 0 is the interface for the cyclic user program on all three CPUs. CPU 1 calls the INITIALIZE function (OB 200) during the cold restart. The connection list is in data block DB 100.

The following blocks must be loaded in the individual CPUs:

| Function        | CPU 1      | CPU 2  | CPU 3      |
|-----------------|------------|--------|------------|
| Restart OB      | OB 20      | —      | —          |
| User program    | FB 0       | FB 0   | FB 0       |
| FB: SEND-DAT    | FB 100     | FB 100 | FB 100     |
| FB: RECV-DAT    | FB 101     | FB 101 | FB 101     |
| Connection list | DB 100     | DB 100 | DB 100     |
| Input DB        | DB 5       | DB 3   | DB 5; DX 4 |
| Output DB       | DB 3; DX 4 | DB 5   | —          |

First of all the connection list (structure described in the section "Data structure") must be written and entered in DB 100:

DB100

 LEN=37 ABS  
 PAGE 1

## — Sub-list 1 —

|      |    |           |                       |
|------|----|-----------|-----------------------|
| 0 :  | KS | =S1       | Send from CPU 1 to .. |
| 1 :  | KY | =001,003; | .. CPU 2 (DB 3)       |
| 2 :  | KY | =002,004; | .. CPU 3 (DX 4)       |
| 3 :  | KY | =000,000; |                       |
| 4 :  | KS | =S2       | Send from CPU 2 to .. |
| 5 :  | KY | =001,005; | .. CPU 1 (DB 5)       |
| 6 :  | KY | =001,005; | .. CPU 3 (DB 5)       |
| 7 :  | KY | =000,000; |                       |
| 8 :  | KS | =S3       |                       |
| 9 :  | KY | =000,000; |                       |
| 10 : | KY | =000,000; |                       |
| 11 : | KY | =000,000; |                       |
| 12 : | KS | =S4       |                       |
| 13 : | KY | =000,000; |                       |
| 14 : | KY | =000,000; |                       |
| 15 : | KY | =000,000; |                       |

## — Sub-list 2 —

|      |    |           |                                 |
|------|----|-----------|---------------------------------|
| 16 : | KS | =S1       | Send from CPU 1 to ..           |
| 17 : | KY | =002,004; | .. CPU 2 (four blocks of data)  |
| 18 : | KY | =003,002; | .. CPU 3 (two blocks of data)   |
| 19 : | KY | =004,000; |                                 |
| 20 : | KS | =S2       | Send from CPU 2 to ..           |
| 21 : | KY | =001,003; | .. CPU 1 (three blocks of data) |
| 22 : | KY | =003,003; | .. CPU 3 (three blocks of data) |
| 23 : | KY | =004,000; |                                 |
| 24 : | KS | =S3       |                                 |
| 25 : | KY | =001,000; |                                 |
| 26 : | KY | =002,000; |                                 |
| 27 : | KY | =004,000; |                                 |
| 28 : | KS | =S4       |                                 |
| 29 : | KY | =001,000; |                                 |
| 30 : | KY | =002,000; |                                 |
| 31 : | KY | =003,000; |                                 |

Data words DW 16 to DW 31 contain the assignment list required for the manual INITIALIZATION function (OB 200). OB 200 is called by the OB 20 shown below in CPU 1 during the restart.

OB 20  
SEGMENT 1

LEN=23 ABS

|      |      |          |                                |
|------|------|----------|--------------------------------|
| 0000 | :L   | KB 2     | MANUAL INITIALIZATION OF       |
| 0001 | :T   | FY 246   | THE PAGES                      |
| 0002 | :    |          |                                |
| 0003 | :L   | KY 1,100 | THE ASSIGNMENT LIST IS ENTERED |
| 0005 | :T   | FW 248   | IN DB 100 FROM DATA WORD 16    |
| 0006 | :L   | KF+16    | ONWARDS                        |
| 0008 | :T   | FW 250   |                                |
| 0009 | :    |          |                                |
| 000A | :L   | KB 246   | SF OB:                         |
| 000B | :JU  | OB 200   | INITIALIZE                     |
| 000C | :    |          |                                |
| 000D | :AN  | F 252.5  | BLOCK END IF THERE IS NO       |
| 000E | :BEC |          | INITIALIZATION CONFLICT        |
| 000F | :    |          |                                |

The error handling routine is inserted here if an initialization conflict occurs (e.g. stop, output message on printer etc.)

0010 :BE

The user program on each CPU is extended by the RECV-DAT and SEND-DAT call. Function block FB 0 shown below is for CPU 1. For the other CPUs, the input parameter CPUN (CPU number) must be modified.

```

FB0                               LEN=31   ABS
SEGMENT 1
NAME:PROG-1

0005      :C   DB100              CONNECTION LIST      DB 100
0006      :JU  FB101              RECEIVE THE INPUT
                                      DATA BLOCKS

0007 NAME :RECV-DAT
0008 CPUN :      KF+1
0009 ERWA :      FY0
000A      :JC  =ERWA              ABORT IF ERROR/WARNING
000B      :
000C      :
```

The cyclic user program is inserted here and reads data from the input data blocks and writes data to the output data blocks.

```

000D      :
000E      :
000F      :
0010      :C   DB100              CONNECTION LIST      DB 100
0011      :JU  FB100              SEND THE OUTPUT
                                      DATA BLOCKS

0012 NAME :SEND-DAT
0013 CPUN :      KF+1
0014 ERWA :      FY0
0015      :JC  =ERWA              ABORT IF ERROR/WARNING
0016      :BEU
0017      :
0018 ERWA :      AFTER ERROR/WARNING
0019      :BE                     EXECUTE ERROR HANDLING
```

The error handling is inserted here, (e.g. stop, output error message on printer or monitor, etc.)

#### REMEMBER



This example (IPC flag extension using function blocks SEND-DAT and RECV-DAT) can only be performed correctly if the special function organization blocks for multiprocessor communication OB 202 to OB 205 are not called outside these function blocks in any of the CPUs.

# SIEMENS

## SIMATIC S5

CPU 946/947

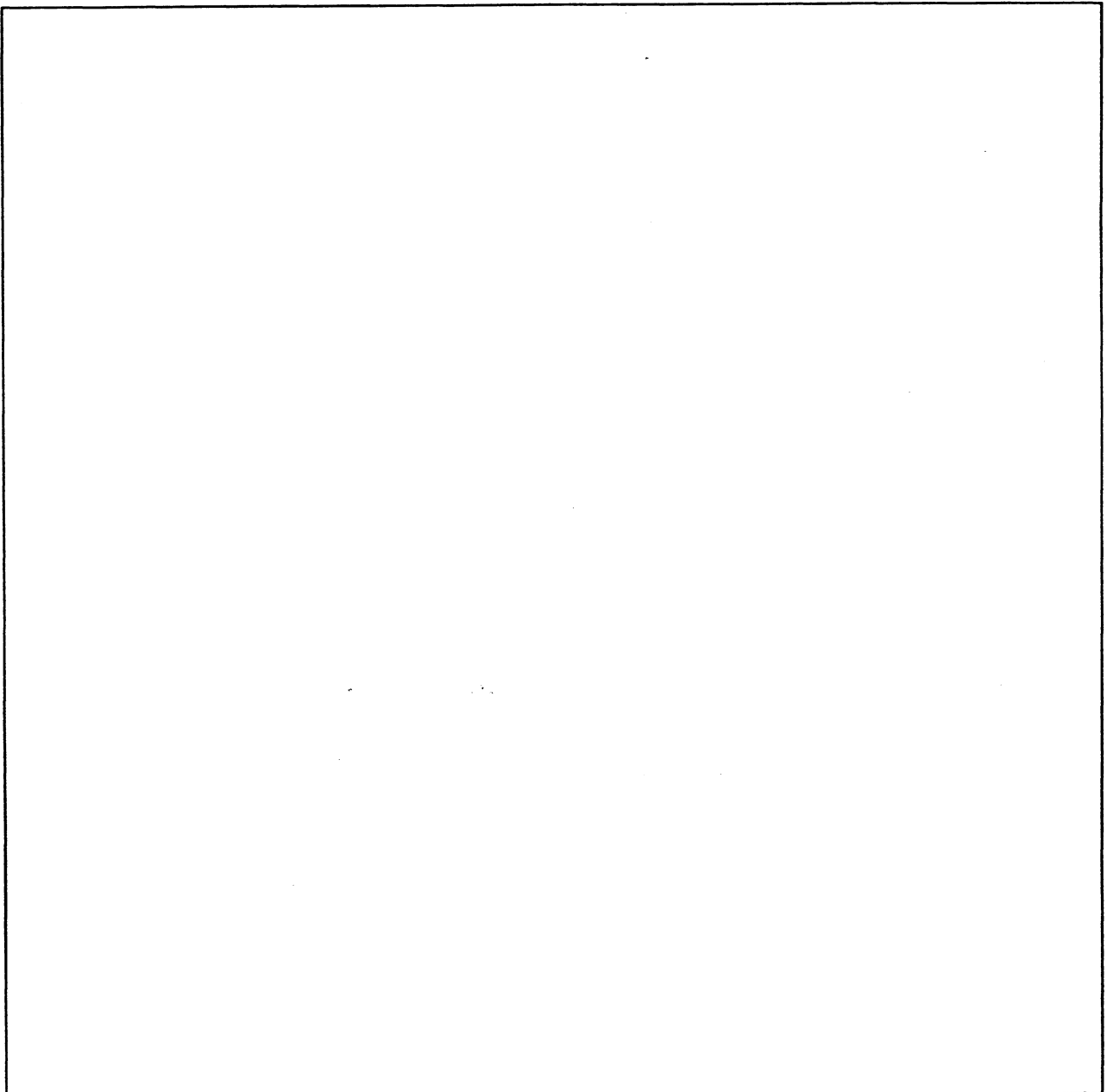
Programmable Controller S5-155U

CPU 946R/947R

Programmable Controller S5-155H

Programming Guide

C79000-B8576-C637-03





---

Siemens has developed this document for its licensees and customers. The information contained herein is the property of Siemens and may not be copied, used, or disclosed to others without prior written approval from Siemens. Users are cautioned that the material contained herein is subject to change by Siemens at any time and without prior notice.

Siemens shall not be responsible for any damages, including consequential damages, caused by reliance on material presented, including but not limited to typographical, electronic, arithmetic, or listing errors.

---

|   |  |
|---|--|
|  |  <b>WARNING</b>   |
|   | <p><b>Hazardous voltage.</b></p> <p><b>Can cause death, severe personal injury, or substantial property damage.</b></p> <p>Restrict use to qualified personnel.<br/>See safety instructions.</p> |

Only qualified personnel should install or maintain this equipment after becoming thoroughly familiar with all warnings, safety notices, and maintenance procedures contained in this manual. The successful and safe operation of this equipment is dependent upon proper handling, installation, operation, and maintenance.

The following are definitions of the terms "qualified person," "danger," "warning," and "caution," as applicable for this document.

#### Qualified Person

One who is familiar with the installation, construction, and operation of this equipment and the hazards involved. In addition, the person should have the following qualifications:

- Be trained and authorized to use and tag circuits and equipment in accordance with established safety practices
- Be trained in the proper care and use of protective equipment in accordance with established safety practices
- Be trained in rendering first aid

#### DANGER

Indicates that loss of life, severe personal injury, or substantial property damage will result if proper precautions are not taken.

#### WARNING

Indicates that loss of life, 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.

---

STEP 5® and SIMATIC® are registered trademarks of Siemens AG.

---

Copyright © Siemens AG 1992

First Printing, September 1989

Printed in the Federal Republic of Germany

---

◆

## Preface

This book contains the programming instructions for the S5-155U programmable controller with CPU 946/947

- 6ES5 946 - 3UA11
- 3UA21
- 3UA22
- 3UA23
- 6ES5 947 - 3UA11
- 3UA21
- 3UA22
- 3UA23

and for the S5-155H programmable controller with CPU 946R,947R

- 6ES5 946 - 3UR11
- 3UR12
- 3UR21
- 6ES5 947 - 3UR11
- 3UR21

If you generate your STEP 5 user program for the S5-155H, please observe the restrictions and notes in the S5-155H manual, part 5, section 2.2.2.

This book is intended for engineers, programmers, and maintenance personnel who have a general knowledge of programmable controller concepts and microcomputer systems.

If you have any questions about CPU 946/947 not answered in this book, please contact your local Siemens representative.

### Important

**For the CPU 946/947 (and CPU 946R/947R) you will require the PG software S5-DOS from V2.0 or S5-DOS/MT.**

**To work with S flags you will require S5-DOS from V3.0 or STEP 5 / MT.**





## How to Use this Book

This information should make working with this Programming Guide easier.

### IMPORTANT

**This guide describes the functions of the S5-155U in multiprocessor operation.**

**If you generate your STEP 5 user program for the S5-155H, please observe the restrictions and notes in the S5-155H manual, Part 5, Section 2.2.2**

### Contents of This Book

- **Chapter 1 - S5-155U Method of Operation and Application**

This chapter describes the S5-155U system structure and discusses it as a further development of the S5-150U programmable controller. It describes the application of CPU 946/947 with its new features and functions. This chapter also discusses the three controller modes of S5-155U operation, the 150U mode and the 155U mode (single and multiprocessor operation).

- **Chapter 2 - User Program**

This chapter describes the STEP 5 programming language and the STEP 5 blocks and how to program them.

- **Chapter 3 - Programming**

This chapter describes cyclic program processing of the CPU 946/947, program organization and program storage.

It explains the STEP 5 operations and provides examples of programs. (You will find more information on STEP 5 operations in the List of Operations to be found in this manual. See also the literature listed under Reference Materials.)

- **Chapter 4 - Modes of Operation**

This chapter explains the modes of operation and program processing levels of the S5-155U programmable controller. It describes cold and warm restart procedures, internal time interrupts, external process interrupts, and hardware (HW) signal interrupts.

- **Chapter 5 - Interrupt and Error Diagnostics**

This chapter discusses error analysis using LEDs, the block stack (BSTACK), the control bits, and the interrupt stack (ISTACK). It also explains handling errors with error organization blocks.

- **Chapter 6 - Integrated Special Functions**

This chapter describes special functions contained in organization blocks as a permanent component of the system program in CPU 946/947.

- **Chapter 7 - Extended Data Block DX 0**

This chapter describes the structure of extended data block DX 0 and its programming. It also provides examples of assigning block parameters to influence the procedure of CPU 946/947.

- **Chapter 8 - Memory Assignment and Memory Organization**

This chapter explains the memory assignment and memory organization of CPU 946/947. It includes the RS registers and their bit assignments for experienced system users and also includes the RS registers for the real-time clock.

- **Chapter 9 - Memory Access Using Absolute Addresses**

This chapter defines local and global memory in the S5-155U programmable controller. It explains how to access the local and global memory areas, including the dual-port RAM area. The chapter discusses operations for accessing registers using ACCU 1, transferring entire memory blocks, and transferring the contents of one register to another. The chapter also presents base address register operations and set operations in the RS/RT area.

- **Chapter 10 - Multiprocessing with Communications Processors (CPs)**

This chapter explains how to configure the S5-155U for multiprocessor operation and explains the application of this programmable controller as a multiprocessing device. This chapter also discusses data exchange between CPUs and between CPUs and CPs. It explains how to program DB1 to allocate peripherals, how to run the start-up procedure for multiprocessing, and how to run the test operation.

- **Chapter 11 - Testing Aids (Online Programmer Functions)**

This chapter describes some special features of online programmer functions as they are related to the S5-155U programmable controller. It explains how to use these functions to test your user program.

- **Chapter 12 - Summary of STEP 5 Operations**

This chapter contains a list of all STEP 5 operations and their permitted parameters.

- **Index**

The index contains an alphabetical list of key terms and subjects covered in this book and their corresponding page numbers.

## **Training**

Contact your local Siemens representative for information on SIMATIC training courses.

### Reference Materials

It is recommended that you have the following books that support the S5-155U system:

- *Catalog ST 54.1: S5-135U, S5-155U and S5-155H Programmable Controllers*  
(Order no. E86010-K4654-A111-A3-7600)\*
  
- Programmer Manuals
- *PG 635 Programmer*  
(Order no. 6ES5 835-0SC21)\*
- *PG 675 Programmer (S5-DOS)*  
(Order no. 6ES5 875-0SC21)\*
- *PG 685 Programmer*  
(Order no. 6ES5 885-0SC21)\*
- *PG 730 Programmer*  
(Order no. 6ES5 834-0FC21)\*
- *PG 750 Programmer*  
(Order no. 6ES5 886-0FC21)\*
- *PG 750-486 Programmer*  
(Order no. 6ES5 886-0FC22)\*
- *STEP 5 Basic Package*  
(Order no. 6ES5 998-0SC21)\*
- *STEP 5 Programming Package for Personal Computers*  
(Order no. 6ES5 896-0SC21)\*
- *S5-135U Programmable Controller (CPU 928)*  
(Order no. 6ES5 998-1UL23)\*
- *S5-135U Programmable Controller (CPU 921/922)*  
(Order no. 6ES5 998-0UL22)\*
- *U Periphery Manual*  
(Order no. 6ES5 998-0PC22)\*
  
- You will find a detailed introduction to programming with STEP 5 and an explanation of the functions of the programmable controller S5-155U and its peripherals in the book

*Automating with the SIMATIC S5-155U*  
by Hans Berger  
Siemens AG, ISBN 3-8009-1562-6

---

\* Order the appropriate book from your local Siemens representative.

● **Appendix: List of Abbreviations**

This appendix lists abbreviations used throughout this book. For an explanation of the special abbreviations of the ISTACK, see section 3.4. Also see the S5-155U List of Operations for definitions.

|                       |   |
|-----------------------|---|
| ACCU 1 (2, 3, 4)-High | High word in accumulator 1 (2, 3, 4), 16 bits                     |
| ACCU 1 (2, 3, 4)-Low  | Low word in accumulator 1 (2, 3, 4), 16 bits                      |
| ACCU 1 (2, 3, 4)-LH   | High byte of the low word in accumulator 1 (2, 3, 4), 8 bits      |
| ACCU 1 (2, 3, 4)-LL   | Low byte of the low word in accumulator 1 (2, 3, 4), 8 bits       |
| ADF                   | Addressing error  |
| BA                    | Block start address   |
| BCD                   | Binary coded decimal number                                       |
| BR                    | Base address register   |
| BSTACK                | Block stack   |
| C                     | Counter   |
| CC 0, CC 1            | Word condition codes  |
| COOR                  | Coordinator module  |
| CP                    | Communications processor  |
| CPU                   | Central processing unit   |
| CSF                   | Control system flowchart method of STEP 5 language representation |
| D                     | Data bit  |
| DB                    | Data block  |
| DBA                   | Data block start address (in register 6)                          |
| DBL                   | Data block length (in register 8)                                 |
| DD                    | Data double word (32 bits)  |
| DL                    | Data byte left (eight bits)                                       |
| DMA                   | Direct memory access  |
| DR                    | Data byte right (eight bits)                                      |
| DW                    | Data word (16 bits)   |
| DX                    | Extended data block   |
| EPROM                 | Erasable programmable read-only memory                            |
| ERAB                  | First bit scanned (bit condition code)                            |
| EU                    | Expansion unit  |
| F                     | Flag bit  |
| FB                    | Function block  |
| FD                    | Flag double word  |
| FW                    | Flag word   |
| FX                    | Extended function block   |
| FY                    | Flag byte   |
| INTAS                 | Interrupt of 80186 microprocessor                                 |
| IP                    | Intelligent input/output module                                   |
| ISTACK                | Interrupt stack   |
| LAD                   | Ladder diagram method of STEP 5 language representation           |
| NAU                   | Power failure   |

|       |   |
|-------|---|
| OB    | Organization block                                      |
| OR    | Or (bit condition code)                                 |
| OS    | Stored overflow (word condition code)                   |
| OV    | Overflow (word condition code)                          |
| OW    | Word from the extended periphery                        |
| OY    | Byte from the extended periphery                        |
|       |   |
| PARE  | Parity error  |
| PB    | Program block   |
| PEU   | Power supply failure in expansion unit                  |
| PG    | Programmer  |
| PI    | Process image   |
| PII   | Process image input table                               |
| PIQ   | Process image output table                              |
| PLC   | Programmable controller                                 |
| PW    | Peripheral word   |
| PY    | Peripheral byte   |
|       |   |
| QVZ   | Timeout   |
|       |   |
| RAM   | Random-access memory                                    |
| RLO   | Result of logic operation (bit condition code)          |
| RS    | System data register                                    |
|       |   |
| S     | S flag  |
| SAC   | STEP address counter                                    |
| SB    | Sequence block  |
| SD    | S flag double word                                      |
| STA   | Status (bit condition code)                             |
| STL   | Statement list method of STEP 5 language representation |
| STS   | Stop operation  |
| SUF   | Substitution error                                      |
| STUEB | BSTACK overflow   |
| STUEU | ISTACK overflow   |
| SW    | S flag word   |
| SY    | S flag byte   |
|       |   |
| T     | Timer   |
| TRAF  | Transfer error  |
|       |   |
| ZYK   | Cycle time exceeded error                               |
|       |   |
| <...> | Contents of register, memory cell etc.                  |

# Contents

|  |      |
|--|------|
| <b>Preface</b> .....   | 0-1  |
| <b>How to Use This Book</b> .....  | 0-3  |
| <b>Chapter 1 - S5-155U Method of Operation and Application</b> .....                 | 1-1  |
| 1.1 Application of CPU 946/947 .....   | 1-8  |
| 1.2 Further Development of the S5-150U Programmable Controller:<br>the S5-155U ..... | 1-9  |
| 1.3 New Features and Functions of CPU 946/947 .....                                  | 1-11 |
| 1.4 Controller Modes of Operation .....  | 1-15 |
| 1.4.1 150U Controller Mode (Single Processor Operation) .....                        | 1-15 |
| 1.4.2 155U Controller Mode (Single Processor Operation) .....                        | 1-16 |
| 1.4.3 155U Controller Mode (Multiprocessor Operation) .....                          | 1-16 |
| <b>Chapter 2 - User Program</b> .....  | 2-1  |
| 2.1 STEP 5 Programming Language .....  | 2-1  |
| 2.1.1 The LAD, CSF, STL, and GRAPH 5 Methods of Representation ..                    | 2-2  |
| 2.1.2 Structured Programming .....   | 2-3  |
| 2.1.3 STEP 5 Operations .....  | 2-4  |
| 2.1.4 Number Representation .....  | 2-5  |
| 2.1.5 STEP 5 Blocks .....  | 2-9  |
| 2.2 Organization, Program, and Sequence Blocks .....                                 | 2-13 |
| 2.2.1 Programming .....  | 2-13 |
| 2.2.2 Calling .....  | 2-13 |
| 2.2.3 Organization Blocks .....  | 2-15 |
| 2.3 Function Blocks .....  | 2-18 |
| 2.3.1 Structure of Function Blocks .....   | 2-19 |
| 2.3.2 Programming Function Blocks .....  | 2-21 |
| 2.3.3 Calling Function Blocks and Assigning Parameters to Them ..                    | 2-25 |
| 2.3.4 Special Function Blocks .....  | 2-28 |
| 2.4 Data Blocks .....  | 2-30 |
| 2.4.1 Structure of Data Blocks .....   | 2-30 |
| 2.4.2 Programming Data Blocks .....  | 2-31 |
| 2.4.3 Opening Data Blocks .....  | 2-32 |
| 2.4.4 Special Data Blocks .....  | 2-35 |
| <b>Chapter 3 - Programming</b> .....   | 3-1  |
| 3.1 Structured Program Overview .....  | 3-1  |
| 3.1.1 Program Organization .....   | 3-1  |
| 3.1.2 Program Storage .....  | 3-6  |
| 3.1.3 Processing the STEP 5 User Program .....                                       | 3-7  |
| 3.1.4 Requirements for Program Use .....   | 3-10 |
| 3.2 STEP 5 Operations Overview .....   | 3-11 |
| 3.3 Supplementary Operations .....   | 3-46 |
| 3.4 System Operations .....  | 3-70 |

|  |         |
|--|---------|
| <b>Chapter 4 - Modes of Operation</b> .....  | 4-1     |
| 4.1 Modes of Operation and Their Program Processing Levels .....                                       | 4-1     |
| 4.2 STOP Mode .....  | 4-6     |
| 4.2.1 Smooth STOP .....  | 4-6     |
| 4.2.2 Hard STOP .....  | 4-11    |
| 4.3 RESTART Mode .....   | 4-12    |
| 4.3.1 Manual and Automatic Cold Restart .....  | 4-13    |
| 4.3.2 Manual and Automatic Warm Restart .....  | 4-14    |
| 4.3.3 Comparison of Cold and Warm Restart .....  | 4-15    |
| 4.3.4 Retentive Cold Restart .....   | 4-16    |
| 4.3.5 Comparison of Cold Restart and Retentive Cold Restart .....                                      | 4-16    |
| 4.3.6 Programming the Start-Up Modes .....   | 4-18    |
| 4.3.7 Interruptions in the RESTART Mode .....  | 4-20    |
| 4.4 RUN Mode .....   | 4-21    |
| 4.4.1 Cyclic Program Processing (Cycle) .....  | 4-22    |
| 4.4.2 Time Driven Program Processing (Internal Time Interrupts) .....                                  | 4-23    |
| 4.4.3 Interrupt Driven Program Processing (External Process Interrupts/<br>HW Signal Interrupts) ..... | 4-28    |
| <br><b>Chapter 5 - Interrupt and Error Diagnostics</b> .....   | <br>5-1 |
| 5.1 Frequent Errors in the User Program .....  | 5-1     |
| 5.2 Error Information Analysis .....   | 5-2     |
| 5.3 Control Bits and Interrupt Stack .....   | 5-5     |
| 5.4 Error Handling Using Organization Blocks .....   | 5-12    |
| 5.5 Causes of Error and Error Organization Blocks .....  | 5-14    |
| <br><b>Chapter 6 - Integrated Special Function Organization Blocks</b> .....                           | <br>6-1 |
| 6.1 Set/Read Time of Day (OB 121) .....  | 6-4     |
| 6.2 Turn Interrupt Disable On/Off (OB 122) .....   | 6-8     |
| 6.3 Delete STEP 5 Blocks (OB 124) .....  | 6-9     |
| 6.4 Generate STEP 5 Blocks (OB 125) .....  | 6-12    |
| 6.5 Transfer Process Image Tables (OB 126) .....   | 6-15    |
| 6.6 Multiprocessor Communication (OB 200, OB 202 to OB 205) .....                                      | 6-19    |
| 6.7 Compare CPU Restart Types in Multiprocessor Operation (OB 223) .....                               | 6-20    |
| 6.8 Copy/Duplicate Data Blocks (OB 254, OB 255) .....  | 6-21    |
| <br><b>Chapter 7 - Extended Data Block DX 0</b> .....  | <br>7-1 |
| <br><b>Chapter 8 - Memory Assignment and Memory Organization</b> .....                                 | <br>8-1 |
| 8.1 Memory Assignment in CPU 946/947 .....   | 8-2     |
| 8.1.1 Memory Assignment for the System RAM .....   | 8-3     |
| 8.1.2 Memory Assignment for the Peripherals .....  | 8-5     |
| 8.2 Memory Organization in CPU 946/947 .....   | 8-8     |
| 8.2.1 Block Headers in User Memory .....   | 8-10    |
| 8.2.2 Block Address List in Data Block DB 0 .....  | 8-11    |
| 8.2.3 RI/RJ Area .....   | 8-12    |
| 8.2.4 RS/RT Area .....   | 8-13    |
| 8.2.5 Addressable System Data Area .....   | 8-35    |



|   |         |
|---|---------|
| <b>Chapter 9 - Memory Access Using Absolute Addresses</b> .....               | 9-1     |
| 9.1 Access to Registers Using ACCU 1 .....                                    | 9-5     |
| 9.2 Transferring Memory Blocks .....  | 9-10    |
| 9.3 Operations with the Base Address Register .....                           | 9-12    |
| 9.4 Operations for Transferring the Contents of One Register to Another ..... | 9-14    |
| 9.5 Accessing the Global Memory .....   | 9-14    |
| 9.5.1 Accessing the Dual-Port RAM Memory .....                                | 9-17    |
| 9.6 Set Operations in the RS/RT Area .....                                    | 9-20    |
| <br>  |         |
| <b>Chapter 10 - Multiprocessing with Communications Processors</b> ..         | 10-1    |
| 10.1 Notes .....  | 10-1    |
| 10.2 Data Exchange between CPUs .....   | 10-3    |
| 10.2.1 Interprocessor Communication Flags .....                               | 10-3    |
| 10.2.2 Multiprocessor Communication .....                                     | 10-7    |
| 10.2.3 Handling Blocks .....  | 10-7    |
| 10.3 Allocation of Peripherals .....  | 10-8    |
| 10.3.1 Data Block DB 1 .....  | 10-8    |
| 10.4 Start-Up Procedure for Multiprocessing .....                             | 10-11   |
| 10.5 Test Operation .....   | 10-12   |
| <br>  |         |
| <b>Chapter 11 - Testing Aids: Online Programmer Functions</b> .....           | 11-1    |
| 11.1 DIRECTORY .....  | 11-2    |
| 11.2 DELETE .....   | 11-2    |
| 11.3 START and STOP .....   | 11-2    |
| 11.4 COMPRESS MEMORY .....  | 11-3    |
| 11.5 STATUS VARIABLES .....   | 11-3    |
| 11.6 STATUS .....   | 11-4    |
| 11.7 FORCE VARIABLES .....  | 11-5    |
| 11.8 FORCE .....  | 11-5    |
| 11.9 PROGRAM TEST .....   | 11-6    |
| 11.10 OUTPUT ADDRESS .....  | 11-9    |
| 11.11 MEMORY CONFIGURATION .....  | 11-9    |
| <br>  |         |
| <b>Chapter 12 - Summary of STEP 5 Operations</b> .....                        | 12-1    |
| <br>  |         |
| <b>Index</b> .....  | Index-1 |

## Figures

|   |      |
|---|------|
| <b>Chapter 1</b>  |      |
| 1.1: Typical S5-155U System Configuration .....   | 1-2  |
| 1.2: Block Diagram of a CPU in the S5-155U Programmable Controller<br>(Multiprocessing) ..... | 1-5  |
| <b>Chapter 2</b>  |      |
| 2.1: Methods of Representation in the STEP 5 Programming Language .....                       | 2-2  |
| 2.2: Storage of Blocks in the Program Memory .....  | 2-11 |
| 2.3: Block Calls that Enable Processing of a Program Block .....                              | 2-14 |
| 2.4: Structure of a Function Block (FB/FX) .....  | 2-19 |
| 2.5: Calling FB 201 and Assigning Parameters to It .....                                      | 2-26 |
| 2.6: Structure of a Data Block .....  | 2-31 |
| 2.7: Validity Area of an Opened Data Block .....  | 2-34 |
| <b>Chapter 3</b>  |      |
| 3.1: Block Nesting Depth and the Block Stack (BSTACK) .....                                   | 3-5  |
| 3.2: Cyclic Program Processing .....  | 3-7  |
| 3.3: Setting (Enabling) a Semaphore in Multiprocessor Operation .....                         | 3-65 |
| <b>Chapter 4</b>  |      |
| 4.1: Modes of Operation and Program Processing Levels .....                                   | 4-2  |
| 4.2: Image of the Interrupted Levels .....  | 4-4  |
| 4.3: Program Processing after Power-on .....  | 4-6  |
| 4.4: Program Processing following Cycle Interruption .....                                    | 4-7  |
| <b>Chapter 7</b>  |      |
| 7.1: Structure of DX 0 .....  | 7-2  |
| <b>Chapter 8</b>  |      |
| 8.1: CPU 946/947 Memory Map .....   | 8-2  |
| 8.2: System RAM Memory Map .....  | 8-4  |
| 8.3: Address Areas for Periphery (8 bits) .....   | 8-5  |
| 8.4: Address Areas for Peripherals/Programming .....  | 8-6  |
| 8.5: Location of Blocks in Memory (Example) .....   | 8-9  |
| <b>Chapter 9</b>  |      |
| 9.1: Local and Global Memory .....  | 9-2  |
| <b>Chapter 10</b>   |      |
| 10.1: Bus Allocation in Multiprocessor Operation .....  | 10-6 |

## Tables

|  |      |
|--|------|
| 4.1: LEDs in Smooth STOP Mode .....                                      | 4-8  |
| 4.2: LEDs in Hard STOP Mode .....  | 4-11 |
| 4.3: LEDs in RESTART Mode during Start-up .....                          | 4-12 |
| 4.4: Comparison of Cold and Warm Restart .....                           | 4-16 |
| 4.5: Comparison of Cold Restart/Retentive Cold Restart .....             | 4-17 |
| 4.6: LEDs in RUN Mode .....  | 4-21 |
| 4.7: Call of Organization Blocks in Time-Driven Program Processing ..... | 4-24 |

# Chapter 1

## S5-155U Method of Operation and Application

This chapter is intended for two different audiences and is divided accordingly as follows:

- Section 1.1 is for first-time users of programmable controllers who are familiar with micro-computer systems.
- Sections 1.2 and 1.3 are for users who are familiar with the S5-150S/S5-150U programmable controller
- Section 1.4 is intended for all users.

### System Structure

A programmable controller is a computer system developed specifically for use in industry (e.g., to control automated manufacturing equipment). Programmable controllers can be modular and consist of a module framework with at least one CPU and a number of peripheral modules. The type and quantity of CPUs and peripheral modules used depend on the automation task to be accomplished.

The S5-155U belongs to the SIMATIC S5 family of programmable controllers. It is the most powerful multiprocessor unit for process automation (open-loop control, signaling, monitoring, closed-loop control, and logging data). Because of its modular construction and very powerful capability, it can control medium-sized to large systems and also handle complex automation tasks at the coordinating and process control level.

You can create your programs with the STEP 5 programming language that has been developed specifically for programmable controllers.

You can configure an S5-155U central controller optionally with the following modules:

- One CPU 946/947 in single processor operation
- Coordinator 923C (COOR 923C) and a maximum of four CPUs (CPU 946/947, 928, 928B, 922, 920) in multiprocessor operation (multi-computing).

The remaining slots in the S5-155U central controller housing are available for communications processors (CPs), intelligent input/output modules (IPs) and input/output modules. You can connect expansion units and the ET 100U to your central controller for additional CPs, IPs, peripherals, and distributed peripherals. Data traffic with the peripheral modules takes place via the parallel S5 bus or via the serial I/O modules. For more information, see the S5-155U Programmable Controller Catalog ST 54.1.

Figure 1-1 shows a typical configuration for an S5-155U programmable controller. This configuration is suitable for multiprocessor operation. The modules outlined in bold suffice for single processor operation.

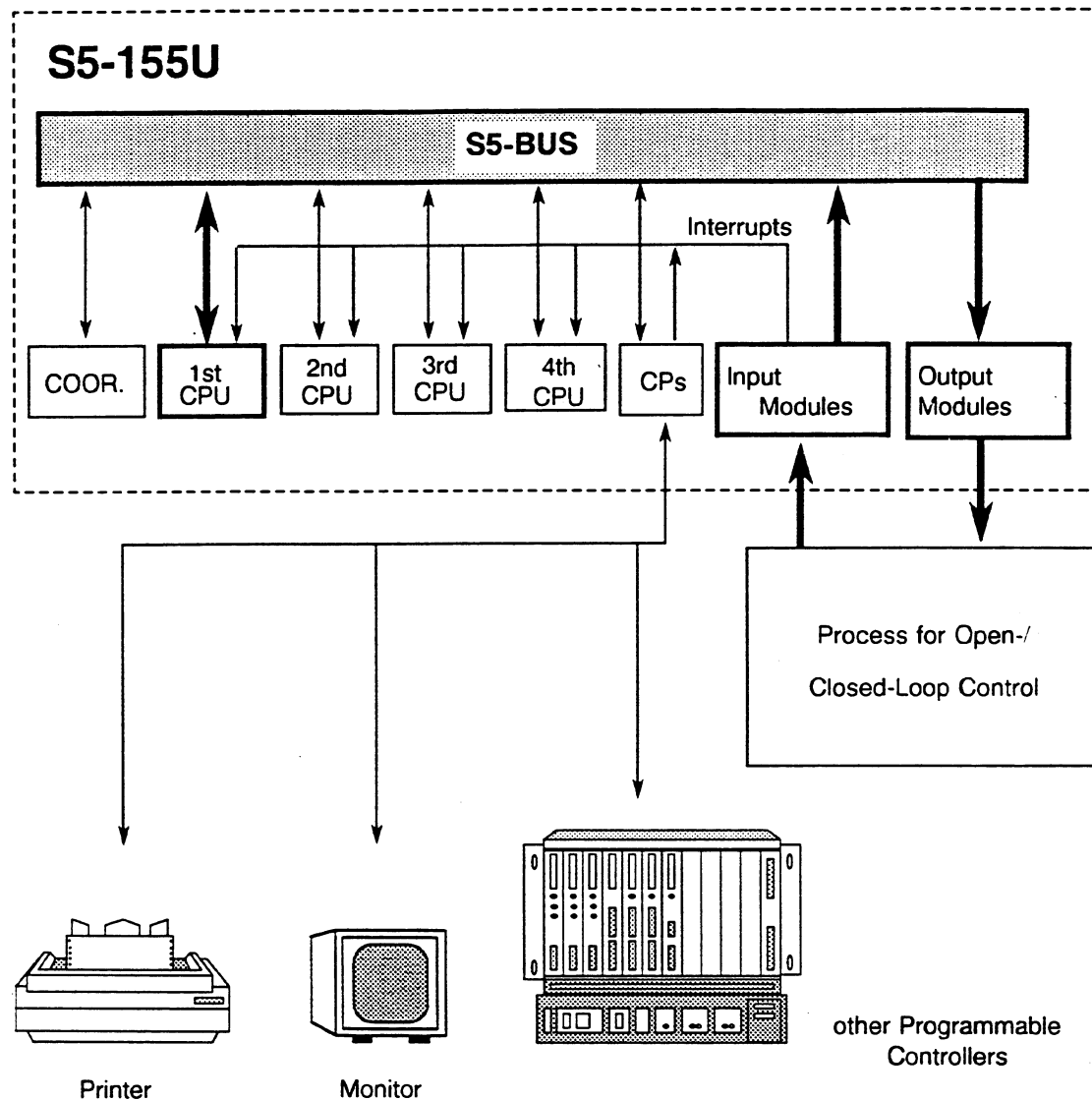


Fig. 1.1 Typical S5-155U System Configuration

Note: If only the CPU 946/947 is used, only 2 CPUs can be plugged in multiprocessor operation.

## **Application**

In single processor operation, CPU 946/947 operates alone. Because of its high speed in processing all operations and its large memory, CPU 946/947 is optimally suited for almost all automation tasks.

For more complex automation tasks, you can expand your S5-155U central controller to a multiprocessing device by using several CPUs at the same time.

Multiprocessing is practical if you want to use special features of different CPUs parallel to each other or if the process you want to control is too extensive for one CPU but can be divided into several independent subtasks. You can assign each task to a CPU that is especially designed to handle it. Each CPU handles its own program independently.

The CPUs access the peripheral modules in sequence via a common backplane bus (the S5 bus). An additional module, the coordinator, grants each CPU access to the S5 bus in sequence in fixed time slices. Only the CPU that has access to the S5 bus can access the peripheral modules. The CPUs can exchange data with each other via the S5 bus. A buffer in the coordinator handles this data traffic.

### Method of Operation

The following cycle repeats constantly within the CPU:

1. All binary input modules assigned to the CPU are scanned and the values that are read in are stored temporarily in the process image input (PII) table.
2. While the user program is processed, values that are to be output are entered in the process image output (PIQ) table.
3. The values contained in the process image output table are output to the output modules assigned to the CPU.

The time that the CPU needs for these three tasks is called the cycle time.

The cycle must run with sufficient speed. The process statuses should not change faster than the CPU can react to them. Otherwise the process would go out of control. You must allow for double the cycle time when figuring the maximum response time. The cycle time depends on the type and scope of the user program and usually is not constant because of conditional block calls and branching in the program.

You can provide an additional time driven program for processes that require control signals in constant time segments. (Up to nine time driven programs are possible with the CPU 946/947 programmable controller.) After a set time period runs out, the cyclic program is interrupted to process the time driven program. The cycle time increases by the time needed to process the time driven program.

You can assign a process interrupt driven program to a process signal that has to trigger an especially fast reaction. (Depending on the mode of operation that you have set, eight external process interrupt driven programs for the 150U mode or four hardware (HW) signal interrupt driven programs for the 155U mode are possible with one CPU 946/947). After an interrupt, the CPU interrupts the cyclic (or time driven) program to process the interrupt driven program. The cycle time increases by the processing time of the interrupt driven program.

In the worst case, the cycle time is a combination of the processing time of the cyclic program and the processing time of the time and interrupt driven programs that might be called up.

The CPU monitors its cycle time. If a programmable limit is exceeded, the CPU interrupts program processing, processes the appropriate user-programmable error organization block, and/or puts itself into the STOP mode and disables the binary outputs.

## Program

The program in the CPU is made up of the user program and the system program.

You create STEP 5 user programs for CPU 946/947 with the STEP 5 programming language, developed specifically for programmable controllers. The user program has a modular structure and consists of at least one program module (block). There are the following two basic block types:

- programmable logic blocks - blocks that contain STEP 5 operations
- data blocks - blocks that contain constants and variables for the STEP 5 program

The system program supports all functions typical of a programmable controller. Such functions include the following:

- Updating the process image (inputs, outputs, interprocessor communication flags)
- Calling cyclic, time, and interrupt driven programs
- Evaluating errors and reacting to them (e.g., timeout, cycle time exceeded, substitution error)
- Managing the memory
- Determining the start-up procedure of the CPU

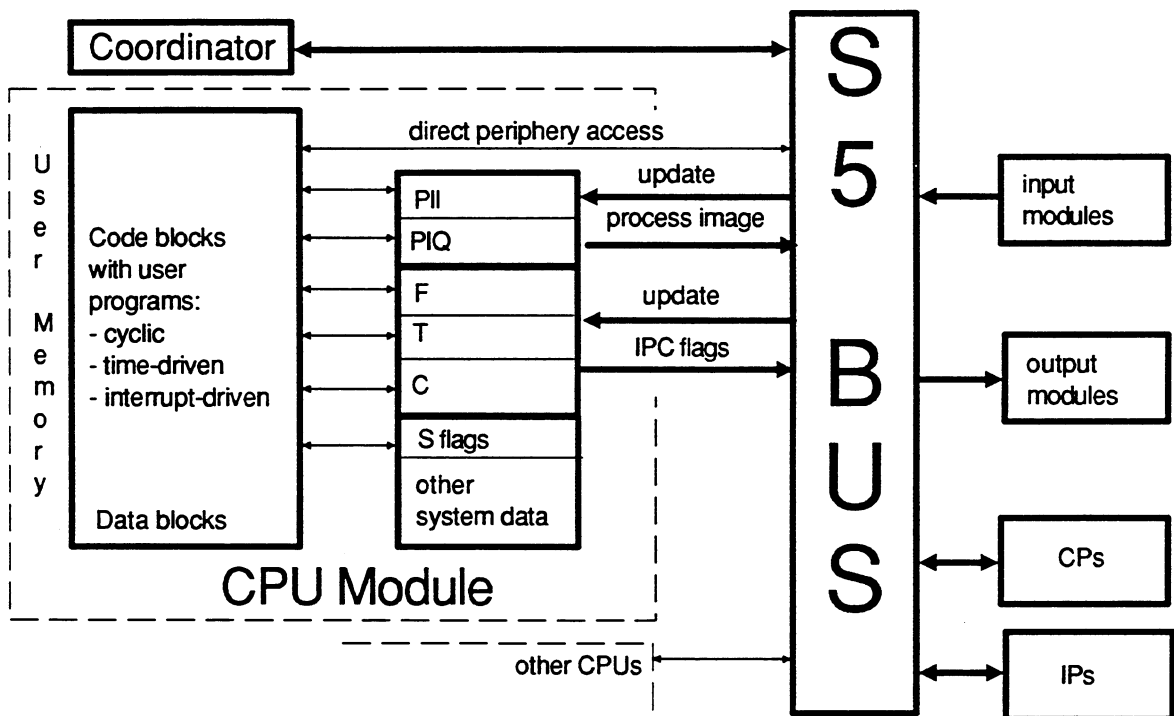


Fig. 1.2. Block Diagram of a CPU in the S5-155U Programmable Controller (Multiprocessing)

## **Address Areas and Registers of CPU 946/947**

The memory of CPU 946/947 is divided into several areas. The following areas are the most important:

- User memory (maximum 896 Kbytes)

The programmable logic and data blocks of the user program are stored in the user memory. The user memory of the CPU 946/947 contains a permanently integrated RAM memory (64 Kwords).

With specified areas for RAM and EPROM memories, the user memory of the CPU 946/947 can be extended with a maximum of six submodules, RAM or EPROM (or a combination of the two) with 64 Kwords each. This gives a maximum memory capacity of 448 Kwords.

- Flag area F (2048 flags)

The flag area is a memory area that the user program can access very fast. Use the flag area for working data that is needed frequently.

The user program can access the following data types: single bits, bytes, words and double words.

You can use individual flag bytes as interprocessor communication flags for data exchange between the CPU and CPs. The system program updates the interprocessor communication flag bytes cyclically via a buffer in the coordinator.

- Extended Flag Area S (32768 S flags)

The CPU 946/947 contains an additional flag area, the S flag area. The user program can access this area in the same way as it accesses the F flags: bit, byte, word, double word. S flags cannot be used as actual operands in function block calls or as interprocessor communication flags for data exchange between the CPUs. PG system software S5-DOS from V3.0 or STEP 5-MT is a requirement for their use.

- Process image input (PII) and output (PIQ) tables (1024 inputs/outputs each)

The user program can access the process image in the same way as it accesses the flag area. The system program updates the process image at the end of a cycle.

- Peripheral area (256 bytes)

The user program can access the peripheral modules directly via the S5 bus. The following data types are possible: bytes and words.

- Timers T (256 timers)

The user program starts timers with a time value between 10 ms and 9990 sec. They are updated according to their time base by the system program.

- Counters C (256 counters)

The user program loads counters with a start value (maximum 999) and counts up or down.



STEP 5 operations (basic operations) can access the following operand areas:

- flag area
- S flag area
- process image input and output tables
- peripheral area
- timers
- counters
- current data block

STEP 5 operations use the following mechanisms to access these operand areas:

- The majority of STEP 5 operations addresses a memory register relative to the beginning of an operand area. As long as the program is working exclusively with these operations, it is separated from the operand areas and cannot overwrite itself if an error occurs.
- A few STEP 5 operations work with absolute addressing. You can access the entire memory area using these operations.

**NOTE:** Because of the extended address area in CPU 946/947, these few STEP 5 operations are not compatible to the S5-150U programmable controller. The same applies to all operating and information functions that work with absolute addressing.

In contrast to other operand areas, the current data block has no fixed start address and length. The current data block is the data block whose start address and length is entered in special registers (see below). The user program can access only the current data block if no operations with absolute addressing are used. The following data types are possible: single bits, bytes, words and double words.

CPU 946/947 has the following registers for processing STEP 5 operations and for buffering data:

- Four accumulators (ACCU 1 to ACCU 4, 32 bits each) that serve as multipurpose registers (e.g., as auxiliary registers for memory-memory transfer or as registers for operands and arithmetic results)
- One operation counter (STEP address counter, SAC) that contains the address of the next operation to be executed
- One data block start address (DBA) register that contains the start address of the current data block
- One data block length (DBL) register that contains the number of data words of the current data block
- One base address (BR) register that you can use to access memory with absolute addresses

### **1.1 Application of CPU 946/947**

The S5-155U programmable controller with CPU 946/947 is the most powerful central controller of the SIMATIC S5 family. It combines the advantages of the S5-150U and the S5-135U programmable controllers for STEP 5 processing and also provides new and expanded functions in the highest range of performance of the SIMATIC S5 family. Its features include the following:

- Fast control and calculation, especially double-word and floating point arithmetic
- Signaling, monitoring, closed-loop control (with standard FB)
- Data logging, communications, operator-interface communication in combination with communications processors (CPs)
- Communication capability also in the smooth STOP mode of the CPU
- Large memory for user programs and user data
- Extended function and extended data blocks (FXs and DXs)
- Multiprocessing
- Processing of four HW signal interrupts (using signal lines of the S5 bus)
- Overall reset using the RESET switch
- Automatic address setting of the memory submodules
- Continuously integrated parity monitoring of the memory
- Integrated real-time clock

See the S5-135U, S5-155U and S5-155H Programmable Controllers Catalog ST 54.1, the CPU 946/947 Hardware and Installation Guide, and the S5-155U List of Operations for more information and technical specifications.

## 1.2 Further Development of the S5-150U Programmable Controller: the S5-155U

This section explains only those points that concern the user and related activities (e.g., test, system start-up, information functions). Only those items that relate to the S5-150U are mentioned, not the new features and functions that are described in the next section. For the corresponding hardware features, see the appropriate hardware and installation guide.

The S5-155U programmable controller is a further development of the S5-150U. However, the two units are not completely compatible. The reason for this is the new physical address space of CPU 946/947, with a total of 448 Kwords (RAM/EPROM).

The S5-150U, on the other hand, has a physical address space of 64 Kwords in which the external memory is accommodated with an additional 64 Kwords accessed via an address window 2 Kwords in length.

As a consequence of the address expansion in CPU 946/947, some STEP 5 operations and all operation and information functions that work with absolute addresses are not compatible. In contrast to this, all STEP 5 operations that work with relative addresses are completely compatible to the S5-150S/U. They are located in the "relative data model".

The "relative data model" separates the programming interface completely from hardware-specific and software-specific implementation on the various S5 central processing units. You work with names (e.g., inputs, outputs, counters) and with relative addresses within the individual operand areas. The absolute address remains in the background.

You can operate CPU 946/947 using S5-150U programs by setting it for the 150U mode of operation. Make this setting in DX0 (see Chapter 7). When using CPU 946/947 with S5-150U programs, note the following differences:

- **For the CPU 946/947 and 946R/947R, you need S5-DOS programmer software, from V2.0.**
- The address space has been extended; the address width is now 20 bits (an address is a double word {32 bits} with the 12 most significant bits padded with zeros).
- The start addresses of all data and peripheral areas in the memory are new. The arrangement of the individual areas has changed to a certain extent.
- The system data area (RS) has been reassigned.
- The register assignment of the CPU has been changed because of the 20-bit address. This affects the STEP 5 operations LIR and TIR.
- The block start address list (in DB 0) is no longer in the user memory. It is in a permanent location in the system data area (register). You no longer have to provide a RAM area.
- The block start address list (in DB 0) no longer contains all the address information. To speed up the run time, only the 16 most significant bits are stored. The CPU pads the four least significant bits with zeros (paragraph addresses).

## *Method of Operation and Application*

---

- Your STEP 5 blocks are no longer located consecutively in the memory. They are stored in the paragraph addresses (i.e., the four least significant bits of the address always equal zero). The resulting gaps are filled with automatically generated fillers. The integrated programmer interface generates the fillers in the RAM area of the programmable controller. The programmer handles this in the EPROM area (select the WORD/FIELD operating mode from the PRESETS menu).
- Operations with absolute addresses run differently or have been replaced by new operations. This affects the STEP 5 operations LIR, TIR, and TNW.
- The bits in the interrupt mask are assigned differently. This affects the STEP 5 operations LIM and SIM.
- The STEP 5 operations IAI, RAI, and TNB are no longer applicable and, consequently, neither are the user interrupts.
- Organization block OB 32 now detects the load error in addition to the transfer error (TRAF). This is a load operation to a data block that has a parameter that is larger than the specified actual block length. The combined error signal is called TLAF.
- You no longer use jumpers to set the start addresses of the individual memories or memory submodules. The system program makes the setting automatically during each cold restart.
- Parity monitoring is integrated in the modules and therefore is always available.
- The programmer interface is integrated and can also be operated centrally via the coordinator.

You will find more details on the operating modes in Section 1.4.



## Method of Operation and Application

---

- Capability to process four HW signal interrupts via signal lines of the S5 bus and user organization blocks (see section 4.4.3)
- Parameter assignment of system performance in extended data block DX 0 (see Chapter 7) is as follows:
  - Capability to select automatic warm restart, automatic cold restart, or manual start-up mode in DX 0.
  - Capability to select one of two different clock distributors for internal time interrupts (fixed cycle, see section 4.4.2).
  - Capability to process nested internal time interrupts (see section 4.4.2).
  - Capability to assign interrupt priority (see section 4.4.3).
- Peripheral address area extended to 64 Kwords (see Chapter 8).  
This extension is available for linearly addressed CPs and IPs and system supplements.
- The COMPRESS MEMORY programmer function is handled in the following new way because of the large memory capacity and the long processing time associated with it:
  - In the STOP mode, after you instruct the PG accordingly, all blocks are shifted (even large data blocks).
  - In the RUN mode, when compressing, large data blocks (larger than 512 words) are not shifted, and so one or more large gaps can result.
- Display of the STS interrupt in the ISTACK if STS caused the CPU to go into the STOP mode (see section 5.3).  
STS is a STEP 5 operation (stop operation).

**Note:** Function blocks with assembler code instead of STEP 5 code cannot run on CPU 946/947. If the STEP 5 operation ASM is processed, the CPU goes into the STOP mode and triggers the SUF substitution error signal in the interrupt stack (ISTACK).

## New Features and Functions from Version -3UA21, 22 onwards

The versions -3UA21 and -3UA22 are identical in their functions. For production reasons the versions were updated to 6ES5946-3UA22 and 6ES5947-3UA22.

CPU 946/947 of the S5-155U has undergone further development. The following new features have been implemented in addition to the functions of S5-DOS programmer software stage II:

- **Multiprocessing (Multi-computing)**  
Operation of several similar or different complete CPUs each with their own memory and with common peripherals on the same system bus (see Chapter 10).  
Integrated special functions are available for communication in multiprocessor operation (see section 10.2.2).

- Additional STEP 5 operations (see sections 3.2 through 3.4), as follows:

Basic operations:

- Load operation for double word (hexadecimal code) L DH

Supplementary operations

- Two-word substitutions DO DW, DO FW, DO =

System operations

- Add operation for double word (hexadecimal code) ADD DH  
(previously ADD DF)
- Load and transfer operations for access to local and global memory areas organized in bytes or words (see Chapter 9) LY/TY GB, LW/TW GW etc.
- Load and transfer operations for access to dual-port RAM pages organized in bytes or words (see Chapter 9) LY/TY CB, LW/TW CW etc.

- New integrated special functions (see Chapter 6), as follows:

|                           |   |
|---------------------------|---|
| OB 124:                   | Delete STEP 5 blocks                                      |
| OB 125:                   | Generate STEP 5 blocks                                    |
| OB 126:                   | Generate process image tables                             |
| OB 200, OB 202 to OB 205: | Functions for multiprocessor communication                |
| OB 223:                   | Compare CPU restart types in multiprocessor operation     |
| OB 254, OB 255:           | Transfer data blocks (DBs) and extended data blocks (DXs) |

- New restart type selectable in extended data block DX 0 - retentive cold restart (see Chapter 7)
- New parameters in extended data block DX 0 for transmission of interprocessor communication (IPC) flags (see Chapter 4 and 7)
- New screen for extended data block DX0 (version 2.0 of the S5-DOS programmer software and higher). This screen offers simple parameter assignment in DX 0 (see Chapter 7).
- The STATUS online programmer function is now possible in the start-up procedure (OB 20, OB 21, OB 22) and in the smooth STOP mode (OB 39) (see section 11.6).

- **The S flags**

To be able to program the S flags you will require the PG software S5-DOS V3.0 or higher and STEP 5/MT.

This area is 32 Kbits in size and is freely available to the user/programmer, i.e. it is not used for standard function blocks or as an IPC flag.

The following operations are available:

|    |    |     |    |        |
|----|----|-----|----|--------|
| A  | S  | 0.0 | to | 4095.7 |
| AN | S  | 0.0 | to | 4095.7 |
| O  | S  | 0.0 | to | 4095.7 |
| ON | S  | 0.0 | to | 4095.7 |
| S  | S  | 0.0 | to | 4095.7 |
| R  | S  | 0.0 | to | 4095.7 |
| =  | S  | 0.0 | to | 4095.7 |
| L  | SY | 0   | to | 4095   |
| L  | SW | 0   | to | 4094   |
| L  | SD | 0   | to | 4092   |
| T  | SY | 0   | to | 4095   |
| T  | SW | 0   | to | 4094   |
| T  | SD | 0   | to | 4092   |

S flags cannot be given as actual operands when function blocks are called.

From the versions

6ES5946-3UA23 and  
6ES5947-3UA23 onwards

also the operands for S-flag operations can be substituted with the "process" operations DO FW, DO DW, DO =, DI and DO RS.



## 1.4 Controller Modes of Operation

CPU 946/947 can operate in the following three controller modes.

Please observe the following points:

- 150U and 155U modes can be set in extended data block DX 0 (see Chapter 7)
- The default is the 150U mode: Multiprocessing is not possible in this controller mode.

### 1.4.1 150U Controller Mode (Single Processor Operation)

The 150U controller mode of CPU 946/947 with single processor operation is the only mode compatible to the S5-150U programmable controller. As explained in section 1.3, compatibility exists where STEP 5 programs remain within the relative data addresses (i.e., they do not work with absolute addresses).

The 150U controller mode is the default setting in the system program. You can also set it in extended data block DX 0 (see Chapter 7).

Note the following relevant points:

- The CPU 946/947 user memory (64 Kwords) is available in its basic structure as an integrated RAM. It is larger than the user memory of the S5-150U programmable controller (48 Kwords). If you operate your system exclusively with RAM, you do not need additional memory modules or submodules.
- However, you can use the complete memory configuration with an optional combination of EPROM/RAM in this mode of operation.
- The following are set as defaults in DX 0.
  - Cyclic (OB 1)
  - processing of up to 8 process interrupts at block boundaries, polling i.e. cyclic scanning of input byte IB 0 (OB 2 to OB 9, see Section 4.4.3).
  - time-driven program processing with up to 9 periods (OB 10 to OB 18, see Section 4.4.2), basic clock rate setting can be selected, nested processing, choice between 2 clock distributors.
  - priority levels for process interrupts and time-driven interrupts possible (DX 0, see Chapter 7).

#### NOTE

**The S5 bus does not process any HW signal interrupts in the 150U controller mode. The CPU does not react to interrupts of bus signals INTA to INTG. Do not enable the HW signal interrupts by altering the hardware.**

- CPU 946/947 does not implement the S5-150U user interrupts.
- The specifications for block nesting depth (maximum 40 blocks) and bracket depth (maximum seven brackets) in the binary area of the S5-150U and CPU 946/947 are the same.

- The program run times and response times of CPU 946/947 are noticeably faster than those of the CPU in the S5-150U.
- Plugging a coordinator into a system set to the 150U controller mode with single processor operation causes the system program to report an error and go into the STOP mode.

### 1.4.2 155U Controller Mode (Single Processor Operation)

Setting the 155U controller mode with single processor operation in extended data block DX 0 and operating CPU 946/947 as a single processor separates the functionality of the S5-155U from the S5-150U. The multiprocessing function of the S5-155U makes it very similar to the S5-135U. External process interrupt processing has been replaced by HW signal interrupt processing. Interrupts occur at operation boundaries. The 155U controller mode is necessary for multiprocessing:

- Cyclic processing (OB 1)
- Processing of 4 interrupts via the S5 bus (OB 2 to OB 5)
  - priority levels within the 4 interrupts possible (DX 0)
  - processing at operation boundaries
- Time-driven processing of up to 9 periods:
  - basic clock rate can be set
  - choice between two clock rate distributors (DX 0)
  - nested processing of time-driven interrupts
  - priority levels between time-driven interrupts and interrupts (DX 0)
  - processing at operation boundaries.

### 1.4.3 155U Controller Mode (Multiprocessor Operation)

For solving complex automation tasks that can be separated technologically, you can operate a maximum of four CPUs parallel to each other in multiprocessor operation.

In addition to the first CPU 946/947, you can operate the following CPUs (see the S5-155U Central Controller Housing Hardware and Installation Guide):

- an additional CPU 946/947 with a maximum of two 355 memory modules
- CPU 928B
- CPU 928
- CPU 922 (R processor)
- CPU 920 (M processor)

In a multiprocessing controller, each CPU processes its own user program independently. You must use a Coordinator 923C for multiprocessing. This module manages the access of the CPUs to peripherals via the S5 bus (arbitration).

Interprocessor communication (IPC) flags are available in the coordinator for exchanging small volumes of data between CPUs. You must define these IPC flags in DB 1 of each CPU (see section 10.3).

**NOTE**

**Plugging a Coordinator 923C into your controller  
sets up multiprocessor operation signaling.**

For multiprocessing, a DB 1 is necessary for each CPU. Each DB 1 should contain the binary input and output bytes allocated to the CPU and the IPC flags. You must also set the 155U controller mode in DX 0.

For more information on multiprocessing, see Chapter 10. Further notes can be found in the book on multiprocessor communication, C79000-B8576-C468.

The details for CPU 946/947 listed in section 1.4.2 also apply to this controller mode of operation.

## Chapter 2 User Program

### 2.1 STEP 5 Programming Language

Use the STEP 5 programming language to convert automation tasks into programs that run on SIMATIC S5 programmable controllers. You can program simple binary functions, complex digital functions, and basic arithmetic operations using STEP 5.

The **operations** of the STEP 5 programming language are divided into the following three sets:

- Basic operations:
  - You can use these operations in all blocks
  - and represent them in ladder diagram (LAD), control system flowchart (CSF), and statement list (STL).
- Supplementary operations:
  - You can use these operations only in function blocks
  - and represent them only in STL.
- System operations:
  - These operations are similar to the supplementary operations.
  - You can use them only in function blocks
  - and represent them only in STL.
  - Only experienced programmers should use system operations.

**2.1.1 The LAD, CSF, STL, and GRAPH 5 Methods of Representation**

When programming in STEP 5, you can choose between the following four methods of representation: LAD, CSF, STL, GRAPH 5. You can choose the method of representation that best suits your programming applications.

The machine code that the programmers generate is the same for all four methods of representation.

*If you follow certain rules when programming in STEP 5, the programmer can translate your program from one method of representation into any other (see the STEP 5 Basic Package manual).*

While CSF, LAD, and GRAPH 5 represent your STEP 5 program graphically, STL represents individual STEP 5 operations with mnemonic abbreviations.

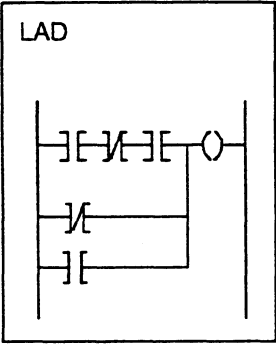
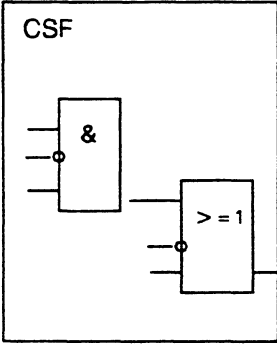
| Ladder Diagram  | Statement List   | Control System Flowchart   |
|---|--|--|
| <p>Programming with Graphic Symbols like a Circuit Diagram</p> <p>corresponds to DIN 19239</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>LAD</p>  </div> | <p>Programming with Mnemonic Abbreviations of Function Designations</p> <p>corresponds to DIN 19239</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>STL</p> <pre style="margin-left: 40px;"> A I AN I A I ON I O I = Q                     </pre> </div> | <p>Programming with Boolean Graphic Symbols</p> <p>corresponds to IEC 117-15<br/>DIN 40700<br/>DIN 40719<br/>DIN 19239</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>CSF</p>  </div> |

Fig. 2.1 Methods of Representation in the STEP 5 Programming Language

GRAPH 5 is an additional method of representation for a programmer to display sequential control graphically. For programs found in the step and transition levels, the programmer is able to convert the LAD, CSF, or STL method of STEP 5 language representation when you enter a convertible program.

### 2.1.2 Structured Programming

The entire program of a CPU consists of the following two parts:

**System program:** It includes all statements and declarations to implement operating functions inside the programmable controller (e.g., protection of data if the power fails and triggering of user reaction to interrupts).

The system program is stored on EPROMs in the CPU 947 and consequently is a permanent part of the CPU. You *cannot* alter the system program.

**User program:** It includes all statements and declarations programmed by you for signal processing. This signal processing affects a controlled system (process) according to a control task. The user program is divided into blocks. After you transfer your program from a programmer, it is stored in the integrated RAM in CPU 947 and in RAM and/or EPROM submodules in the 355 memory module.

Divide your program into individual, self-contained program sections (blocks). The division of your program clarifies the essential program structures at a glance or emphasizes system parts that are related throughout 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 program test section by section (block by block),
- 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 the following two types of blocks: programmable logic blocks - they contain STEP 5 statements for signal processing and include organization blocks (OBs), program blocks (PBs), function blocks (FBs), sequence blocks (SBs), and data blocks (DBs) - they contain data, variables, and constants.



### 2.1.4 Number Representation

Depending on the operations to be carried out, the following number representations are permitted in STEP 5:

- Binary numbers:
- a) 16-bit fixed-point numbers
  - b) 32-bit fixed-point numbers
  - c) 32-bit floating-point numbers
- Decimal numbers:
- d) numbers in BCD code

When you use a programmer to input or display number values, set the data format on the programmer (main menu) in which you would like to enter or display the values (e.g., KF for fixed-point). The programmer converts the internal number representation to a form that you can read directly.

You can carry out all arithmetic operations with the 16- and 32-bit fixed-point numbers, including comparison, addition, subtraction, multiplication, and division.

Use numbers in BCD code only for input and output. You *cannot* carry out any arithmetic operations with them directly.

Use 32-bit fixed-point numbers to execute comparison operations. Addition (new: +D) and subtraction (new: -D) operations are also available. The 32-bit fixed-point numbers are also necessary as an intermediate level for converting numbers in BCD code to floating-point numbers and for calculating addresses.

The STEP 5 programming language also has **conversion operations** that enable you to convert numbers directly to other number representations.



## 16-bit and 32-bit Fixed-Point Numbers

Fixed-point numbers are whole binary numbers that have a sign.

- Representation in a programmable controller as follows:

Fixed-point numbers are 16 bits (one word) or 32 bits (two words) wide. Bit 15 or bit 31 contains the sign: 0 indicates a positive number and 1 indicates a negative number.

Negative numbers are represented in their two's complement representation.

32-bit, fixed-point number:



- When you enter a fixed-point number at a programmer:

indicate the data format at the programmer for 16-bit fixed point numbers:

KF

indicate the data format for 32-bit fixed point numbers:

only DH

Permissible number range:     -32768 to +32767             (16 bit)

8000 0000H to 7FFF FFFFH (32 bit)

-2147483648 bis +2147483647

Use fixed-point numbers for simple calculations and for comparing number values.

## Floating-Point Numbers

Floating-point numbers are positive and negative fractional numbers.

- They are represented in a programmable controller as follows:

Floating-point numbers 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 eight bits long.

The exponent indicates the order of magnitude of the floating-point number. The sign of the exponent tells you whether the amount of the floating-point number is larger or smaller than 0.1.

The mantissa indicates the accuracy of the floating-point number as follows:

Accuracy with a 24-bit mantissa:  $2^{-24} = 0.000000059604$

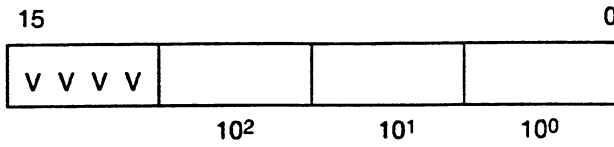
(corresponds to seven decimal places)



### Numbers in BCD Code

Decimal numbers are represented as numbers in BCD code. With sign and three digits, they occupy 16 bits (one word) in an accumulator, as shown in the following example:

|          |          |        |        |
|----------|----------|--------|--------|
| Bit No.  |          |        |        |
| 15 to 12 | 11 to 8  | 7 to 4 | 3 to 0 |
| Sign     | Hundreds | Tens   | Units  |



The individual digits are positive 4-bit binary numbers between 0000 and 1001 (0 and 9).

The permissible data range is: -999 to +999

The left bits are reserved for the sign as follows:

|                             |        |
|-----------------------------|--------|
| sign for a positive number: | "0000" |
| sign for a negative number: | "1111" |

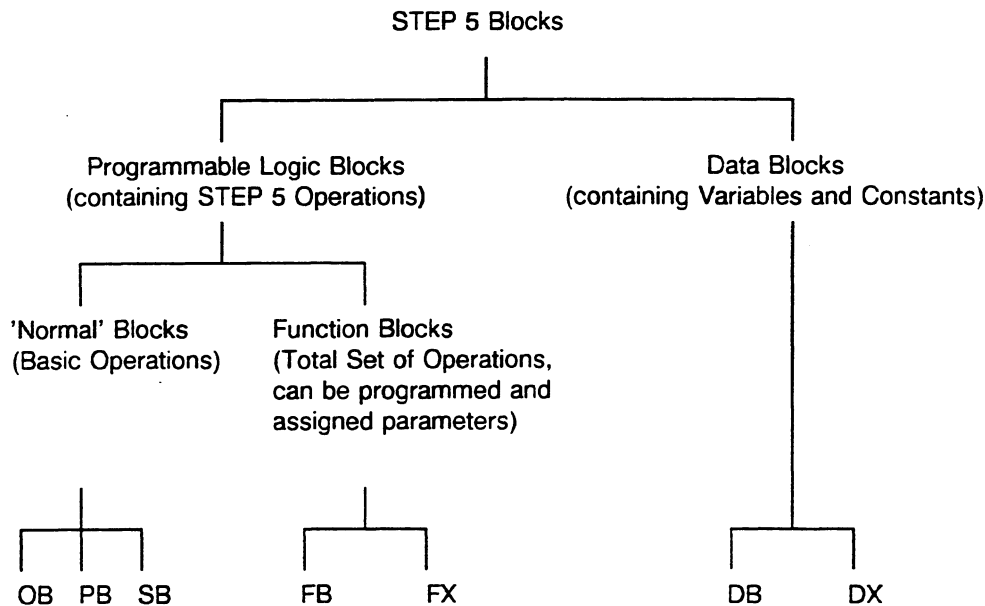
### 2.1.5 STEP 5 Blocks

A block is a part of the user program distinguished by function, structure, or application.

The following characteristics identify a block:

- Block type - organization block (OB), program block (PB), sequence block (SB), function block (FB), extended function block (FX), data block (DB), extended data block (DX)
- Block number - a number between 0 and 255

STEP 5 blocks are classified as shown below:



The STEP 5 programming language differentiates among the following block types:

- Organization blocks (OBs)  
Organization blocks are the interfaces between the system program and the user program. They can be divided into the following three groups:

OB 1 to OB 39 - called by the system program. They control program processing, the start-up procedure of the CPU, and reaction in the event of an error. You, the user, program these blocks.

OB 40 to OB 120 - system program blocks that you cannot call.

OBs with a number greater than 120 are the integrated special functions. They are a permanent component of the system program that you can call as needed in your user program.

- Program blocks (PBs)

Use program blocks to structure your program. They contain program parts divided according to technological or functional criteria. Program blocks generally contain the greatest part of your program.

- Sequence blocks (SB)

Sequence blocks are special program blocks used for step-by-step processing of sequential cascading. Sequence cascades can now also be programmed via GRAPH 5. Sequence blocks in STEP 5 therefore no longer have the same significance. Sequence blocks can be used in the same way as program blocks. Thus the number of program blocks available increases from 256 to 512.

- Function blocks (FB/FX)

Use function blocks and extended function blocks to program frequently recurring and/or complex functions (e.g., digital functions, sequence control, closed-loop control, signaling functions). A function block can be called several times by higher-order blocks and supplied with new operands (assigned parameters) at each call.

- Data blocks (DB/DX)

Data blocks and extended data blocks contain the data with which the user program works. This data can be permanent or variable. This type of block contains no STEP 5 statements and has a distinctly different function from the other blocks.

### A block is structured as follows:

All blocks consist of two parts:     - a block header and  
   - a block body.

The **block header** is always five data words long. The programmer always stores the following information in the block header:

- block start ID
- block type
- block number
- programmer ID
- library number
- block length (including header)

Block header in the  
program memory

|                                       |                |
|---------------------------------------|----------------|
| Start                                 | ID             |
| Block Type                            | Block Number   |
| Programmer ID                         | Library Number |
| Library Number                        |                |
| Block Length Including Header (Words) |                |

15

0

The **block body** contains - independent of the block type

- STEP 5 operations (in OB, PB, SB)
- variable or constant data (in DB, DX)
- formal operand list + STEP 5 operations (in FB, FX)

The OB, PB, SB, FB, and FX block types are terminated with the STEP5 block end operation BE.

The programmer also generates a **block preheader** (DV, DXV, FV, FXV) for block types DB, DX, FB, and FX. These block preheaders contain information on the data format (for DB and DX) or on the jump labels (for FB and FX). Only the programmer can evaluate this information. Consequently the block preheaders are not transferred to the programmable controller memory. You have no direct influence on the contents of a block preheader.

A STEP 5 block can take up a maximum of 4096 words in the program memory of the CPU.

The following block types are available in CPU 946/947:

- OB 1 to OB 39
  - PB 0 to PB 255
  - SB 0 to SB 255
  - FB 0 to FB 255
  - FX 0 to FX 255
  - DB 2 to DB 255
  - DX 2 to DX 255
- } 512 in total
- } 508 in total

Data blocks DB 0 and DB 1 and extended data blocks DX 0 and DX 1 are reserved for specific functions. You cannot use them.

The programmer stores all programmed blocks in the program memory in random order (see below). The program memory is in a 64 Kword RAM in the CPU in basic configuration and in RAM and/or EPROM submodules in two 355 memory modules for each CPU 946/947. The start addresses of all stored blocks are placed in an address list in data block DB 0 in a permanent area of the system memory.

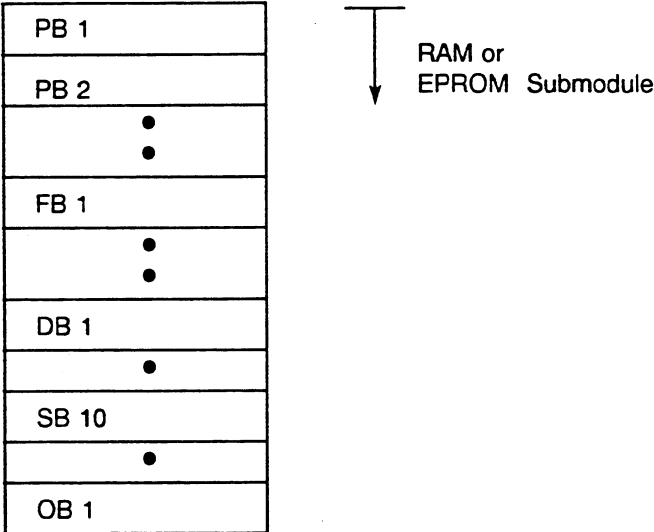


Fig. 2.2 Storage of Blocks in the Program Memory

## *User Program*

---

When operating the CPU with RAM submodules, note the following points:

When blocks are corrected, old blocks in the memory are declared invalid and new blocks are entered into the memory. Similarly, when blocks are deleted, they are not really deleted. Instead, they are declared invalid. Gaps in the memory that develop from deleted blocks are reused when you load new blocks.

You can use the COMPRESS MEMORY programmer function to delete these gaps of old blocks in the memory to make room for new blocks. This function pushes all valid blocks together.

If you use the COMPRESS function while your programmable controller is in the STOP mode, all blocks are shifted. However, if your programmable controller is in the RUN mode, long DBs/DXs (i.e., more than 512 data words) cannot be shifted because of data consistency. Consequently, with COMPRESS in the RUN mode, relatively large open data areas result. You can use these areas for loading large blocks.

## 2.2 Organization, Program, and Sequence Blocks

Organization, program, and sequence blocks are the same with respect to programming and calling. You can program all three types in the LAD, CSF, and STL methods of representation.

### 2.2.1 Programming

When programming organization, program, and sequence blocks, proceed as follows:

- 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:

|                     |          |
|---------------------|----------|
| organization blocks | 1 to 39  |
| program blocks      | 0 to 255 |
| sequence blocks     | 0 to 255 |

- Enter your program in STEP 5 programming language.

#### NOTE

**When programming OBs, PBs, and SBs, you can use only the STEP 5 basic operations.**

- Terminate your program input with the block end operation BE.

#### NOTE

**A STEP 5 block should always contain the terminating operation BE. Logic operations must always be terminated within a block.**

### 2.2.2 Calling

You must enable blocks to process them. Use block calls to enable them (see Figure 2-3).

You can program block calls inside an organization, program, function, or sequence block. They are similar to jumps to subroutines. Each jump causes a block change.

The following applies to a block jump: during a jump to a block that has been called and during the return to the block where the call originated, registers are maintained.



## User Program

Block calls can be unconditional or conditional, as follows:

- Unconditional call: JU xy (x = block type, e.g. FB; y = block no.)

The addressed block is processed *regardless* of the previous result of logic operation (RLO).

The RLO is the signal state in the CPU that is used for further processing of binary signals. For example, the RLO can be combined logically with the signal state of operands, or operations are executed depending on the previous RLO.

The JU jump statement belongs to the unconditional operations. It has no effect on the RLO. The RLO is carried along with the jump to a new block. There it can be evaluated but no longer combined logically.

- Conditional call: JC xy (x = block type, e.g. FB; y = block no.)

The JC jump statement belongs to the conditional operations. The addressed block is processed only if the previous RLO equals 1. If the RLO equals 0, the jump statement is not executed. However, in this case, the RLO is set to 1.

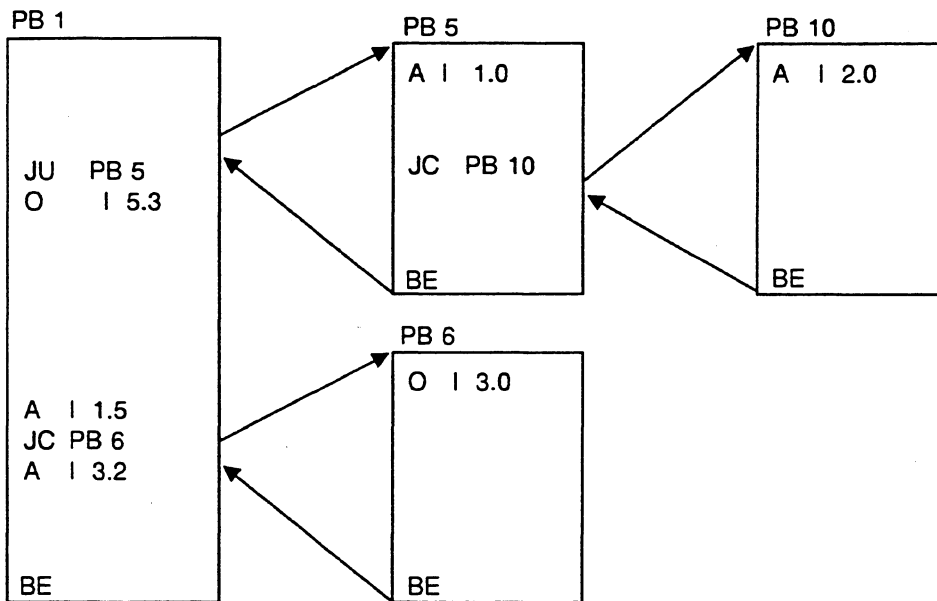


Fig. 2.3 Block Calls that Enable Processing of a Program Block

After the BE or BEC (when RLO equals 1) statement, program processing jumps back to the block in which the block call was programmed. Program processing continues with the first STEP 5 statement that follows the initial block call.

The BE block end statement is processed 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. The result can be evaluated there.

### 2.2.3 Organization Blocks

Organization blocks form the interfaces between the system program and the user program. Organization blocks OB 1 to OB 39 are parts of the user program that you program, just like program, function, and sequence blocks. By programming these OBs, you can influence the behavior of the CPU during start-up, program processing, and in the event of an error.

For this purpose, there are the following four classes of organization blocks:

- OBs for organizing normal operation of cyclic, time driven, or interrupt driven program processing
- OBs for organizing the start-up procedure
- OBs for reacting to errors
- OBs for communication in the smooth STOP mode

In addition, the CPU 946/947 contains special function organization blocks, see Chapter 6.

Organization blocks are effective as soon as you load them into the memory of the programmable controller. You can also load them into the memory when the programmable controller is running. The system program calls these OBs as a reaction to certain events.

On the next page you will find an overview of the user organization blocks in CPU 946/947 and references to parts of this book where you can find more information on these blocks. The table also includes a description of the function of these organization blocks and of each event that triggers their calls.

| Organization Block  | Function and Call Criteria  | Section              |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
|---|---|----------------------|------------------------|----------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|-------|-------|--------|--------|-------|--------|--------|--------|---------|--------|--------|---------|---------|--------|---------|---------|-------|
| <b>OBs for Program Processing Levels</b>                                      |   |                      |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| OB 1  | Organization of cyclic program processing. Call only after the end of a start-up type and in open continuous cycle.   | 4.4.1                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| OB 2<br>OB 3<br>OB 4<br>OB 5<br>OB 6<br>OB 7<br>OB 8<br>OB 9                  | <b>Only in the 150U Controller Mode:</b><br>Organization of interrupt driven program processing. Call caused by signal state change in input byte "IB 0" (process interrupt) by the following bits:<br><br>I 0.0<br>I 0.1<br>I 0.2<br>I 0.3<br>I 0.4<br>I 0.5<br>I 0.6<br>I 0.7   | 4.4.3                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| OB 2<br>OB 3<br>OB 4<br>OB 5  | <b>Only in the 155U Controller Mode</b> (can be set in extended data block DX 0):<br>Organization of interrupt driven program processing. Call via interrupt lines of the S5 bus:<br><br>Interrupt signal X (INT A or INT B, depending on slot)<br>Interrupt signal INT E<br>Interrupt signal INT F<br>Interrupt signal INT G   | 4.4.3                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| OB 10<br>OB 11<br>OB 12<br>OB 13<br>OB 14<br>OB 15<br>OB 16<br>OB 17<br>OB 18 | Organization of time driven program processing (internal time interrupt) basic clock rate (basic setting T = 100 ms) and clock distributor (basic setting corresponds to S5-150U) in extended data block DX 0. Call in each case with:<br><br><table border="1" data-bbox="455 1395 1125 1745"> <thead> <tr> <th data-bbox="455 1395 637 1455">Basic Setting</th> <th data-bbox="637 1395 871 1455">150U Clock Distributor</th> <th data-bbox="871 1395 1125 1455">2n Clock Distributor</th> </tr> </thead> <tbody> <tr> <td data-bbox="455 1455 637 1484">0.1 s</td> <td data-bbox="637 1455 871 1484">T x 1</td> <td data-bbox="871 1455 1125 1484">T x 1</td> </tr> <tr> <td data-bbox="455 1484 637 1513">0.2 s</td> <td data-bbox="637 1484 871 1513">T x 2</td> <td data-bbox="871 1484 1125 1513">T x 2</td> </tr> <tr> <td data-bbox="455 1513 637 1541">0.5 s</td> <td data-bbox="637 1513 871 1541">T x 5</td> <td data-bbox="871 1513 1125 1541">T x 4</td> </tr> <tr> <td data-bbox="455 1541 637 1570">1.0 s</td> <td data-bbox="637 1541 871 1570">T x 10</td> <td data-bbox="871 1541 1125 1570">T x 8</td> </tr> <tr> <td data-bbox="455 1570 637 1599">2.0 s</td> <td data-bbox="637 1570 871 1599">T x 20</td> <td data-bbox="871 1570 1125 1599">T x 16</td> </tr> <tr> <td data-bbox="455 1599 637 1628">5.0 s</td> <td data-bbox="637 1599 871 1628">T x 50</td> <td data-bbox="871 1599 1125 1628">T x 32</td> </tr> <tr> <td data-bbox="455 1628 637 1656">10.0 s</td> <td data-bbox="637 1628 871 1656">T x 100</td> <td data-bbox="871 1628 1125 1656">T x 64</td> </tr> <tr> <td data-bbox="455 1656 637 1685">20.0 s</td> <td data-bbox="637 1656 871 1685">T x 200</td> <td data-bbox="871 1656 1125 1685">T x 128</td> </tr> <tr> <td data-bbox="455 1685 637 1714">50.0 s</td> <td data-bbox="637 1685 871 1714">T x 500</td> <td data-bbox="871 1685 1125 1714">T x 256</td> </tr> </tbody> </table> | Basic Setting        | 150U Clock Distributor | 2n Clock Distributor | 0.1 s | T x 1 | T x 1 | 0.2 s | T x 2 | T x 2 | 0.5 s | T x 5 | T x 4 | 1.0 s | T x 10 | T x 8 | 2.0 s | T x 20 | T x 16 | 5.0 s | T x 50 | T x 32 | 10.0 s | T x 100 | T x 64 | 20.0 s | T x 200 | T x 128 | 50.0 s | T x 500 | T x 256 | 4.4.2 |
| Basic Setting   | 150U Clock Distributor  | 2n Clock Distributor |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 0.1 s   | T x 1   | T x 1                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 0.2 s   | T x 2   | T x 2                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 0.5 s   | T x 5   | T x 4                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 1.0 s   | T x 10  | T x 8                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 2.0 s   | T x 20  | T x 16               |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 5.0 s   | T x 50  | T x 32               |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 10.0 s  | T x 100   | T x 64               |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 20.0 s  | T x 200   | T x 128              |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| 50.0 s  | T x 500   | T x 256              |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |
| OB 38<br><br>OB 39  | Organization of the start-up procedure for communication in the smooth STOP mode.<br><br>Organization of the cyclic program for communication in the smooth STOP mode.  | 4.2.1                |                        |                      |       |       |       |       |       |       |       |       |       |       |        |       |       |        |        |       |        |        |        |         |        |        |         |         |        |         |         |       |

| Organization Block                               | Function and Call Criteria   | Section                            |     |
|--|--|------------------------------------|-----|
| <b>OBs for Program Processing Levels (Cont.)</b> |  |                                    |     |
| OB 20  | Call upon request for cold restart (manual or automatic after power is restored, see DX 0) | 4.3                                |     |
| OB 21  | Call upon request for manual warm restart/retentive cold restart                           |                                    |     |
| OB 22  | Call upon request for automatic warm restart/retentive cold restart                        |                                    |     |
| <b>OBs for Error Reactions</b>                   |  |                                    |     |
|  | Call Criteria  | Reaction when OB is not programmed | 5.4 |
| OB 19  | Call of a block that is not loaded   | None                               |     |
| OB 19  | Attempt to open a DB/DX data block that is not loaded                                      | STOP mode                          |     |
| OB 23  | Timeout during direct access to digital /analog peripherals                                | None                               |     |
| OB 24  | Timeout during update of the process image   | None                               |     |
| OB 25  | Addressing error   | STOP mode 1)                       |     |
| OB 26  | Cycle time exceeded  | STOP mode                          |     |
| OB 27  | Substitution error   | STOP mode                          |     |
| OB 28  | Timeout input byte IB 0 (process interrupts, 150S/U controller mode)                       | STOP mode                          |     |
| OB 29  | Timeout, distributed peripherals, extended address sets                                    | None                               |     |
| OB 30  | Timeout and parity error in the user memory  | STOP mode                          |     |
| (OB 31)  | (Set cycle time) 2)  | None                               |     |
| OB 32  | Load/transfer error during access to data blocks   | STOP mode                          |     |
| OB 33  | Collision of time interrupt errors   | see 4.4.2                          |     |
| OB 34  | Error at G DB/GX DX  | STOP mode                          |     |

1) This is only relevant if the addressing error is not disabled by the STEP 5 operation IAE (see list of Operations).

2) In new programs, use DX 0 instead of OB 31.

After the system program has called the appropriate organization block, the user program contained in that block is processed. The user program can also call these organization blocks for testing purposes (JU/JC OB xx). However, it is not possible to trigger a cold restart by calling OB 20, for example, because OB 20 is a system program implemented only at start-up and cannot be accessed during user cyclic program processing.

You can assign parameters in data block DX 0 for other system functions (see Chapter 7).

Special function OBs can be called without restrictions (see Chapter 6).

*Calling the system program OBs (OBs greater than 39) in the user program is illegal.*

## 2.3 Function Blocks

Function blocks (FBs) and extended function blocks (FXs) are parts of the user program just like program blocks. FXs have the same structure as FBs and are programmed the same way.

Use function blocks to implement frequently recurring or very complex functions.

As compared to organization, program, and sequence blocks, function blocks have the following four essential **differences**:

- You can assign **parameters** to function blocks. This means that different actual operands can replace the formal operands of a function block with each call. In this way, function blocks that have been created for a general application have various uses.
- You can program function blocks with the entire operation set of the STEP 5 programming language. This set includes the basic operations, which can be used in all types of blocks, and the **supplementary** and **system operations**.

### NOTE

**You can program supplementary and system operations only in function blocks.**

- You can program and document function blocks **only in the STL** method of representation.

However, you can *call* function blocks in the CSF and LAD methods of representation. They are represented graphically as boxes.

- You can give function blocks a **title** with a maximum of eight characters.

Within the user program, each function block represents a complex, complete function.

- You can program function blocks yourself

or

- you can order them from Siemens as standard function blocks (package with diskette and documentation). You can use these standard function blocks to generate user programs quickly and consistently for open-loop control, signaling, closed-loop control, and data logging.

### Note:

Standard function blocks occupy function blocks FB 1 to FB 199 as well as a few data blocks.

### 2.3.1 Structure of Function Blocks

The **header** (five words) of a function block has the same structure as the headers of the other block types.

However, the **body of a function block** has a different structure than the bodies of the other block types. The block body contains the actual program of the function block. In this program, the function is written in the STEP 5 programming language in the statement list (STL) method of representation. *Between* the block header and the actual STEP 5 user program, a 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 over with an unconditional jump that the programmer generates automatically. This jump statement is not displayed on the programmer.

You can enter operands in a function block absolutely (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.

When a function block is called, only the block body is processed.

Fig. 2.4 shows the structure of a function block in the memory of a programmable controller.

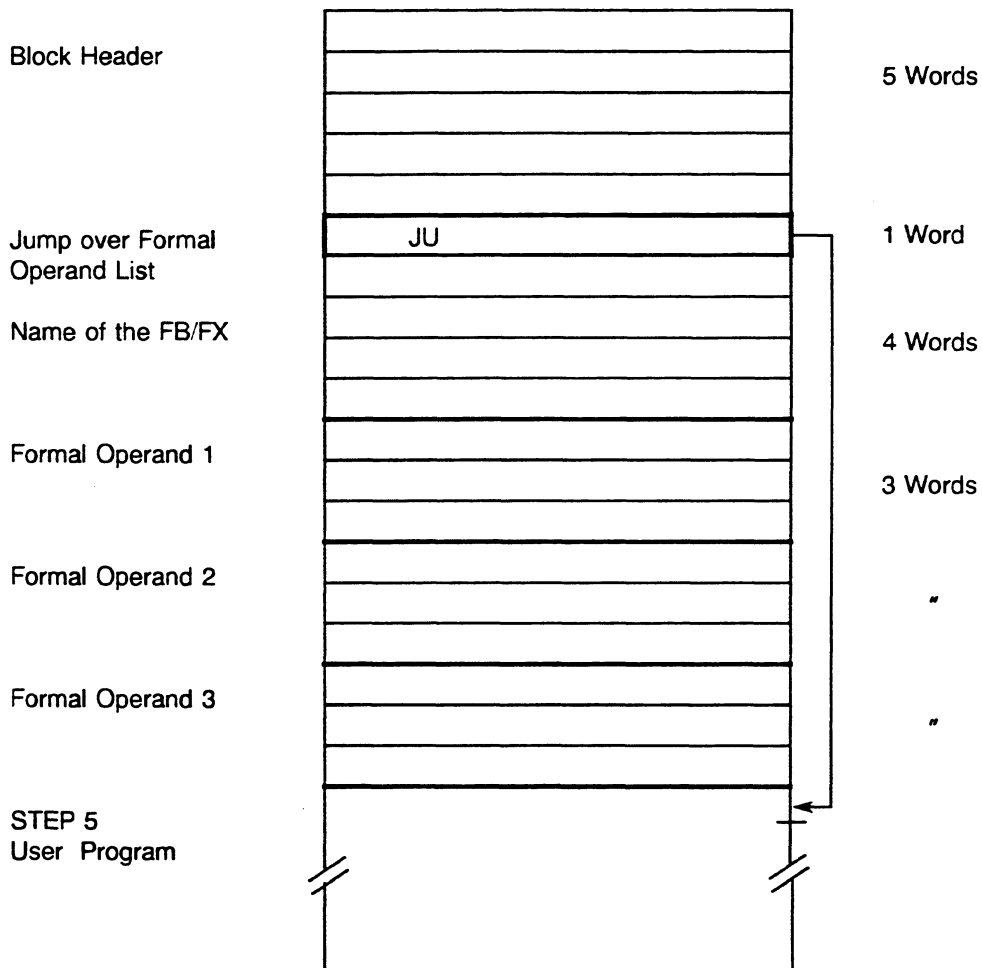


Fig. 2.4 Structure of a Function Block (FB/FX)

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 an incorrect input.

### NOTE

When handling function blocks, distinguish between the following procedures:

- a) Programming FB/FX
- b) Calling FB/FX and then assigning actual values to the parameters

During programming, you specify the function of the block. The operands that you enter are formal operands that perform a place-holding function.

Of course, you can also program all operations with real operands.

During call of a block using a higher order block (OB, PB, SB, FB, FX), the actual operands replace the formal operands. The formal operands are parameters assigned to the function block. The operation and the operand are brought together only during processing of FB/FX.

The following pages provide details on programming function blocks and assigning parameters to them.

### 2.3.2 Programming Function Blocks

When entering a function block at a programmer, perform the following steps:

- Enter the **number** of the function block.

#### NOTE

**Number your user 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. All numbers from 0 to 255 are available for extended function blocks.**

- Assign the function block a **library number** from 0 to 99999. This number is optional. The library number is assigned to a function block independent of its block number or its name. It is intended for your cataloging purposes and is insignificant for the CPU and the programmer.

This same library number can also be used for other blocks.

- Enter the **name** of the function block. The name can have a maximum of eight characters and must begin with a letter. This name is also only important for your documentation purposes.
- Enter the **formal operands** that you are going to use in the function block (maximum of 40 formal operands).

Enter the following information for each formal operand:

1. name of the block parameter
2. type of the block parameter
3. data type of the block parameter, if it is requested by the PG.

The **name** can have a maximum of four characters.

The programmer provides you with the following selection of **block parameter types**:

I = input parameter  
Q = output parameter  
D = date  
B = block operation  
T = timer  
C = counter

When you call an FB/FX in LAD or CSF, then I, D, B, T, and C are parameters that are indicated to the left of the function symbol in graphic representation. Parameters designated with Q are indicated to the right of the function symbol.



## User Program

You must also indicate one of the following data types for **block parameter types I, Q, and D**:

BI/BY/W/D                                    for parameter type I or Q  
 KM/KH/KY/KS/KF/KT/KC/KG            for parameter type D

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).

| Parameter Type | Data Type   | Actual Operands Permitted  |
|----------------|---|--|
| I, Q           | <p>BI for an operand with bit address</p> <p>BY for an operand with byte address</p> <p>W for an operand with word address</p> <p>D for an operand with double word address</p> | <p>I    x.y    input</p> <p>Q    x.y    output</p> <p>F    x.y    flag</p> <p>IB    x    input byte</p> <p>QB    x    output byte</p> <p>FY    x    flag byte</p> <p>DL    x    data byte left</p> <p>DR    x    data byte right</p> <p>PY    x    peripheral byte</p> <p>IW    x    input word</p> <p>QW    x    output word</p> <p>FW    x    flag word</p> <p>DW    x    data word</p> <p>PW    x    peripheral word</p> <p>ID    x    input double word</p> <p>QD    x    output double word</p> <p>FD    x    flag double word</p> <p>DD    x    data double word</p> |
| D              | <p>KM for a binary pattern (16 bits)</p> <p>KY for two absolute numbers, one byte each, each in the range from 0 to 255</p>   | Constants  |

| Parameter Type | Data Type  | Actual Operands Permitted   |
|----------------|--|---|
| D              | KH for a hexadecimal pattern with a maximum of four digits<br>KS for two alphanumeric characters<br>KT for a time (in BCD code) with time base .0 to .3 and time 0 to 999<br>KC for a count (in BCD code) 0 to 999<br>KF for a fixed-point number - 32768 to + 32767<br>KG for a floating-point number <sup>1)</sup> | Constants   |
| B              | Data type designation not possible   | DB x Data blocks; the operation C DBx is executed.<br>FB x Function blocks (permitted only without parameters) are called unconditionally (JU ..x).<br>OB x Organization blocks are called unconditionally (JU ..x)<br>PB x Program blocks are called unconditionally (JU..x).<br>SB x Sequence blocks are called unconditionally (JU ..x). |
| 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.

## User Program

---

- Enter your STEP 5 program in the form of a statement list.

The formal operands are indicated by an equals sign that precedes them (e.g., A = X1). They can also be referenced more than once at various positions in the function block.

### NOTE

- If you change the order or the number of formal operands in the formal operand list, you must also update the following accordingly: Substitution commands in the STEP 5 program of the function block, block parameter list in the block where the call originated.
  - Program or change function blocks only on diskette or hard disk (Winchester) and transfer them to your CPU afterward.
- Terminate your program input with the block end command BE.

### Example: Programming a function block

FB 202

NAME: **EXAMPLE**

|                    |                       |                      |                     |
|--------------------|-----------------------|----------------------|---------------------|
| DECL : <b>INP1</b> | I/Q/D/B/T/C: <b>I</b> | BI/BY/W/D: <b>BI</b> |                     |
| DECL : <b>INP2</b> | I/Q/D/B/T/C: <b>I</b> | BI/BY/W/D: <b>BI</b> | Formal Operand List |
| DECL : <b>OUT1</b> | I/Q/D/B/T/C: <b>Q</b> | BI/BY/W/D: <b>BI</b> |                     |

: **A** = **INP1**  
: **A** = **INP2**  
: **=** = **OUT1**  
: **BE**

Formal  
Operand

Parameter  
Type

Data  
Type

STEP 5 Program

### 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. Although the STEP 5 program is always written in STL, you can call function blocks in graphic representation (CSF or LAD), as well as in a statement list.

This is how you call and assign parameters to a function block:

#### NOTE

**Before calling a function block and assigning parameters to it, you must have already programmed it and transferred it to the program diskette or entered it directly into the program memory of your CPU.**

- When calling a function block and assigning parameters to it, 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.

The call can be either unconditional or conditional as follows:

- Unconditional call (JU FBn for function blocks FB or BA FXn for extended function blocks FX):

The function block that is called is processed regardless of the previous RLO.

- Conditional call (JC FBn for function blocks FB or BAB FXn for extended function blocks FX):

The function block that is called is processed only if the previous RLO equals 1. If the RLO equals 0, the jump operation is not carried out. However, this operation will set the RLO to 1.

After an unconditional or conditional call, the RLO cannot be combined logically. However, during the jump, it is carried over to the function block that is referenced and can be evaluated there.

After you enter the call statement (e.g., JU FB 200), the name of the function block in question and the formal operand list appear automatically. Proceed as follows:

- Assign the actual operand that applies to this call to each of the individual 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 can assign a maximum of 40 actual operands for each call of a function block.

## User Program

---

When program processing jumps to the function block, the actual operands from the block where the call originates are used instead of the formal ones when the function block program is processed.

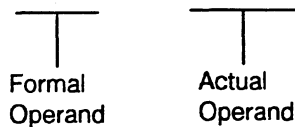
This feature of function blocks that enables you to assign them parameters provides you with programming versatility.

### Example: Calling FB 201 and Assigning Parameters to It using STL and CSF/LAD in a Program Block

- STL method of representation

PB 25

```
      : JU   FB 201
NAME: REQUEST
DATA : DW 1
RST  : I 3.5
SET  : F 2.5
TIME : T 2
TIME : KT010.1
TRAN : DW 1
BEC  : Q 2.3
LOOP : Q 6.0
```



- CSF/LAD method of representation

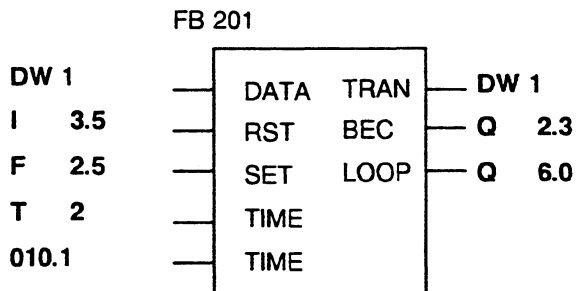


Fig. 2.5 Calling FB 201 and Assigning Parameters to it

The following **example** is intended to further clarify the programming and calling of a function block and the assignment of parameters to it.

Programming FB 202

FB 202  
**SEGMENT 1**

NAME: **EXAMPLE**

|                    |                       |                      |         |
|--------------------|-----------------------|----------------------|---------|
| DECL : <b>INP1</b> | I/Q/D/B/T/C: <b>I</b> | BI/BY/W/D: <b>BI</b> | Formal  |
| DECL : <b>INP2</b> | I/Q/D/B/T/C: <b>I</b> | BI/BY/W/D: <b>BI</b> | Operand |
| DECL : <b>OUT1</b> | I/Q/D/B/T/C: <b>Q</b> | BI/BY/W/D: <b>BI</b> | List    |

```

: A = INP1
: A = INP2
: = = OUT1
: BE
    
```

Formal  
 Operand

Parameter  
 Type

Data  
 Type

STEP 5 Program

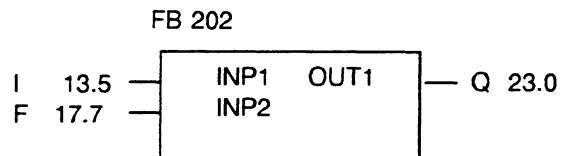
Calling FB 202 and Assigning Parameters to It

- STL method of representation
- CSF/LAD method of representation

```

PB 25
:JU FB 202
NAME :EXAMPLE
INP1 : I 13.5
INP2 : F 17.7
OUT1 : Q 23.0
    
```

Formal Operand      Actual Operand



The following program is executed after program processing jumps to FB 202:

```

: A I 13.5
: A F 17.7
: = Q 23.0
: BE
    
```

### 2.3.4 Special Function Blocks

- **Standard function blocks**

In addition to the function blocks that you program yourself, you can order standard function blocks as a finished software product. They contain standard functions for general use (e.g., signaling functions and sequence control). Note the following restrictions if you use standard function blocks (SFBs):

- The numbers FB 1 to FB 199 are assigned to standard function blocks. Therefore you should use only the numbers FB 200 to FB 255 for your function blocks.
- Flag bytes 200 to 255 are assigned to standard function blocks. You cannot use these. Therefore, you can start with FB0 and go up through FB 199.
- Timer 0 and counter 0 are assigned to standard function blocks.
- Data word DW 0 in data blocks is assigned to standard function blocks.
- Under certain circumstances, you must reserve blocks DB 2, DB 3, and DB 4 for closed-loop control functions.
- Sequence blocks SB 0, SB 2, and SB 3 for GRAPH 5 are assigned to standard function blocks.

If you order standard function blocks, note the special instructions in the accompanying description (e.g., concerning assigned areas and conventions).

Standard Function Blocks and Driver Software for Programmable Controllers of the U-Range Catalog ST 57 lists the standard function blocks for CPU 946/947 with their run time, memory requirements, and assigned variables.

#### **Example of a standard function block**

Floating-point root extractor RAD:GP

The floating-point root extractor RAD:GP is an example of a standard function block. This function block 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 off.

For the rest of the processing, the function block sets the radicand negative ID.

Number range:

Radicand -0,1469368 exponent -38 to +0,1701412 exponent +39  
 Root +0,3833434 exponent -19 to +0,1304384 exponent +20

Function:  $Y = \sqrt{A}$   
 Y = SQRT; A = RAD

Call of the Function Block:

- STL method of representation

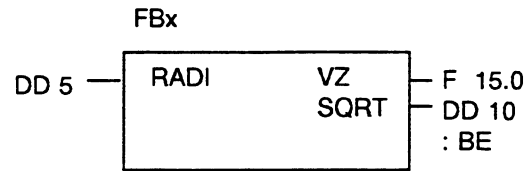
```

Seg-      : C DB 17
ment 1    :
          : ***
          : JU FB x
          : NAME : RAD : GP
Seg-      : DD 5
ment 2 *) : VZ   : F 15.0
          : SQRT : DD 10
    
```

DD = data double word

- LAD/CSF method of representation

SEGMENT 2



\*) Must stand in separate segments as the "CDB 17" operation cannot be translated into LAD/CSF in segment 1.

In the above example, the root is being extracted from a floating-point number that is provided in DD 5 of the DB 17 with an 8-bit exponent and a 24-bit mantissa. The result, another 32-bit, floating-point number, is stored in DD 10. Prior to this, the appropriate data block must be opened. The VZ parameter (parameter type Q, data type BI) indicates the sign of the radicand: VZ equals 1 for a negative radicand. Flag words FW 238 to FW 254 are occupied in this SFB.

Note: Function blocks programmed with assembler code instead of STEP 5 code **cannot** run in CPU 946/947. If the STEP 5 operation ASM is processed, the CPU goes into the STOP mode and triggers the SUF substitution error signal in the interrupt stack (ISTACK).



## 2.4 Data Blocks

Data blocks (DBs/DXs) 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 can include the following:

- optional 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 logging texts)

### 2.4.1 Structure of Data Blocks

A data block consists of the following parts:

- block preheader (DV, DXV)
- block header
- block body

The **block preheader** is set up automatically. It contains the data format of the data words entered into the block body. You have no influence over the setup of the block preheader.

The **block header** occupies five words in the memory and includes the following:

- block ID
- programmer ID
- block number
- library number
- block length (including the length of the block header)

The **block body** contains the data words with which the user programs works. These data words are in ascending order in the block body, starting with data word DW 0. Each data word takes up one word (16 bits) in the memory.

A data block can take up to 2000 words (max. 4096 words inclusive header) in the CPU memory. When you use your programmer to enter and transfer data blocks, take into consideration the size of its memory.

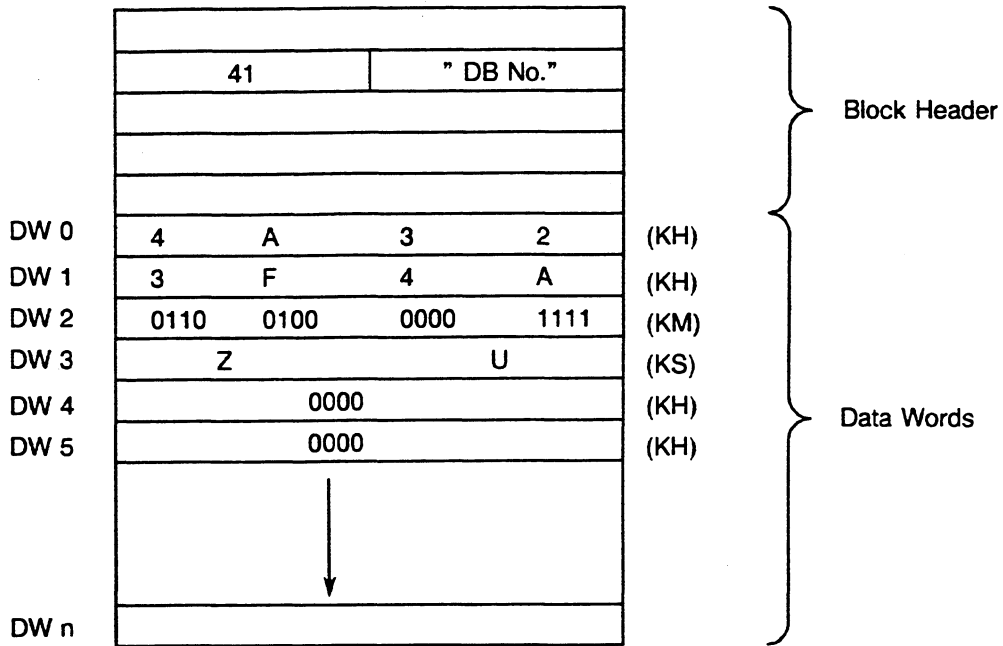


Fig. 2.6 Structure of a Data Block

### 2.4.2 Programming Data Blocks

Create a data block as follows:

- Enter a data block number between 2 and 255 for DBs and DXs.

**NOTE**

Data blocks DB 0 and DB 1 and extended data blocks DX 0 and DX 1 are reserved for specific functions and therefore cannot be used optionally (see section 2.4.4).

- Enter individual data words in the data format you want.

| Permissible Data Format    | Example           |
|----------------------------|-------------------|
| KM = bit pattern           | 00100110 00111111 |
| KH = hexadecimal           | 263F              |
| KY = byte                  | 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               |

**NOTE**

Do not terminate entry of data words with the block end statement BE.

### 2.4.3 Opening Data Blocks

You can open a data block (DB/DX) unconditionally only. This is possible within an organization, program, sequence, or function block. You can open a specific data block more than once in a program.

Open a data block as follows:

- Open DBs with the statement           C DBxx.
- Open DXs with the statement       CX DXxx.

#### NOTE

**Data blocks DB 0 and DB 1 and extended data blocks DX 0 and DX 1 cannot be opened by the user program.**

All statements following with the operand area D refer to the opened block after a data block has been opened.

The opened data block remains valid even if program processing is continued in a different block via a jump statement.

If a different data block is opened in this block, this data block is only valid in the called block. After returning to the called block the old data block is valid again.

You can **access** the data stored in the opened data block during program processing using **load and transfer operations**:

- A **load operation** places the contents of the data word number that is referenced into ACCU 1.

The following are load commands:

|        |               |
|--------|---------------|
| L DLxx | (left byte)   |
| L DRxx | (right byte)  |
| L DWxx | (word)        |
| L DDxx | (double word) |

- A **transfer operation** transfers data from ACCU 1 to the data word number that is referenced.

The following are transfer commands:

|        |
|--------|
| T DLxx |
| T DRxx |
| T DWxx |
| T DDxx |

Loading does not change the contents of a data word.

Transferring overwrites the old contents of a data word.

**NOTE**

Before accessing a data word, you must open the data block you want in your program. This is the only way that the CPU can find the correct data word. The address of the data word number must be available in the opened DB/DX (address is less than or equal to the DB/DX length). Otherwise, the system detects a load/transfer error (TLAF) and calls OB 32. If OB 32 is not programmed, the CPU goes into the STOP mode.

With STEP 5 load and transfer operations, you have access only up to data word number 255. If you want to access larger data blocks, see Chapter 9.

**Example 1: Transferring of data words**

Transfer the contents of DW 1 from data block DB 10 to DW 1 of data block DB 20.

Enter the following statements:

C DB 10 (Open DB 10.)  
L DW 1 (Load DW 1 into the accumulator 1.)  
C DB 20 (Open DB 20.)  
T DW 1 (Transfer DW 1 from the accumulator 1 to DW 1 in DB 20.)

**Example 2: Validity area of current data blocks**

Open data block DB 10 (C DB 10) in program block PB 7. Program processing then processes the data of this data block.

Program processing executes PB 20 after it calls this block (JU PB 20). However, data block DB 10 is still valid. The data area does not change until program processing opens data block DB 11 (C DB 11). Data block DB 11 is valid until the end of program block PB 20 (BE).

Data block DB 10 is valid again after program processing returns to program block PB 7.

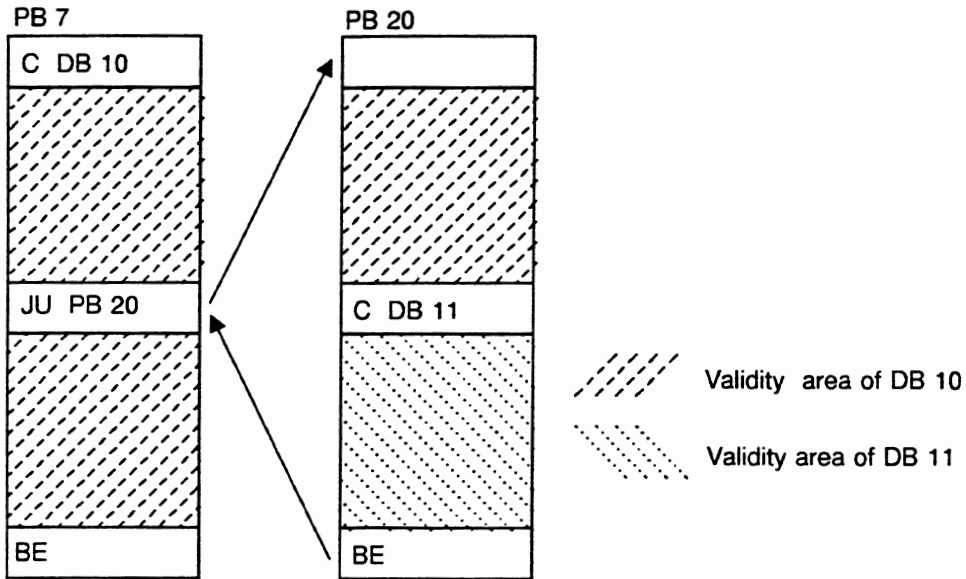


Fig. 2.7 Validity Area of an Opened Data Block

#### 2.4.4 Special Data Blocks

Data blocks DB 0 and DB 1 and extended data blocks DX 0 and DX 1 are reserved for special functions. The system program manages them. They are not optionally available to you.

- **Data Block DB 0** (see section 8.2.2)

This data block contains the address list with the start addresses of all blocks that are located in the user memory of the CPU. The system program generates this address list during a cold restart and checks it during a warm restart. The system program updates this list automatically when you use a programmer to change data blocks or when you generate a data block using the G DB/GX DX operation.

- **Data block DB 1** (see section 10.3.1)

This data block contains the list of digital inputs/outputs ("P" peripherals with relative byte addresses from 0 to 127) and the interprocessor communication flag inputs/outputs that are assigned to the CPU.

For multiprocessing, you must create a DB 1 for each CPU in your controller. DB 1 is not necessary for single processor operation. However, you can use it under certain circumstances to diminish the cycle time since only the inputs and outputs specified in DB 1 are updated.

- **Extended Data Block DX 0** (see Chapter 7)

This extended data block contains the presetting information of specific system program functions (e.g., during start-up processing). You can change this presetting information in DX 0 to adapt the performance of the system program to your needs.

- **Extended Data Block DX 1**

This extended data block is used in the CPU 946/947 with the COM 155H software on the programmer.

## Chapter 3 Programming

### 3.1 Structured Program Overview

You can process your STEP 5 program in various ways.

- Cyclic program processing is most common. This means that program processing passes through organization block OB 1 cyclically according to instruction of the system program. Via various block calls, the system program processes the user program managed in OB 1 from the beginning to the block end operation BE of OB 1.
- The CPU 946/947 offers in addition time-driven processing (9 levels)
- and process-interrupt processing (4 or 8 levels, according to the mode).

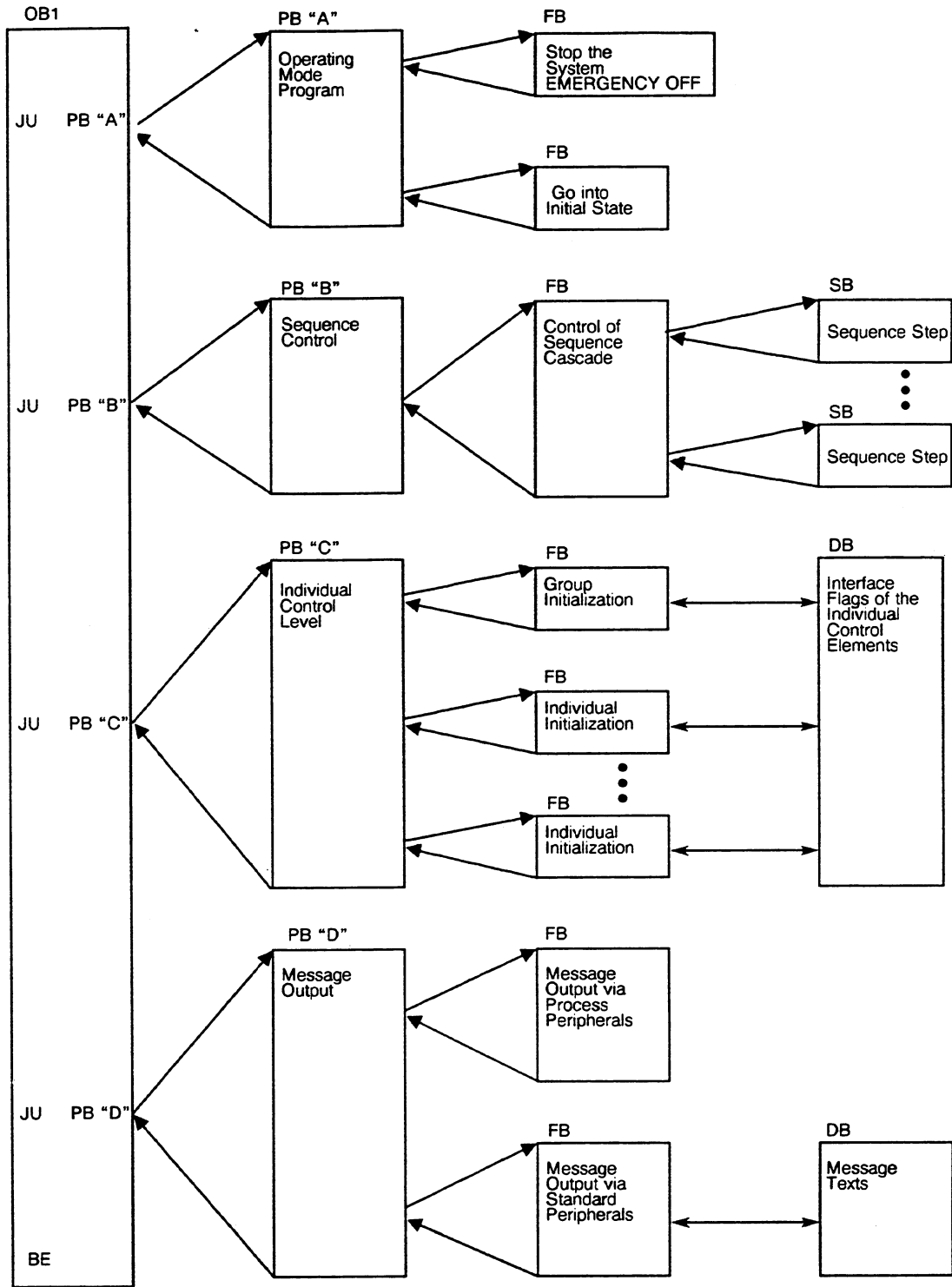
#### 3.1.1 Program Organization

Program organization enables you to specify the processing sequence and level for the blocks you have created. Organize your program by programming organization blocks with conditional or unconditional calls of the blocks you want.

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 them in series or nested in one another.

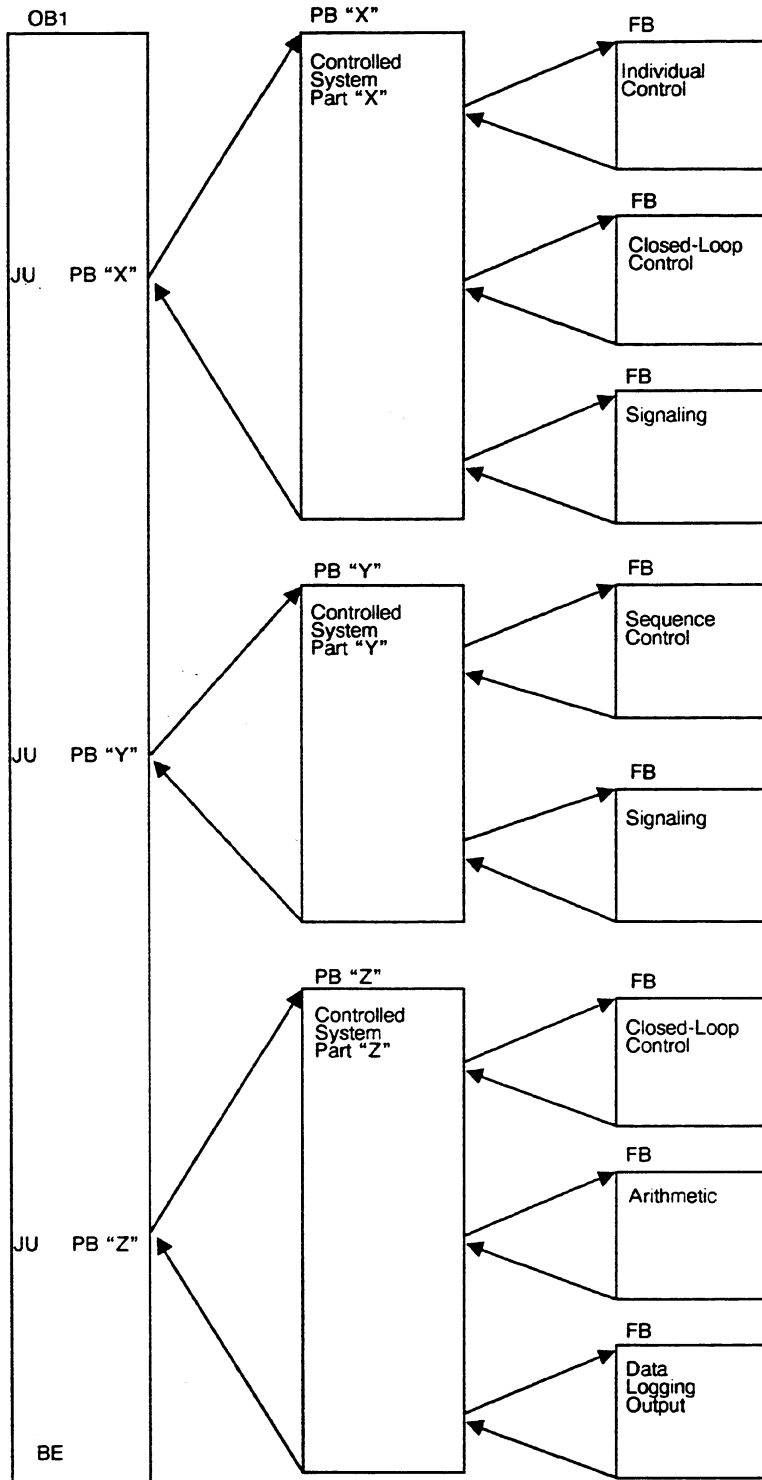
Organize your program to emphasize essential program structures or software-related controlled system parts.

Example: Organization of the User Program: Program Structure with OB 1





Example: Organization of the user program according to controlled system structure



**NOTE**

**You can nest a maximum of 40 blocks in your program. If you nest more than 40 blocks, the CPU reports an error and goes into the STOP mode.**

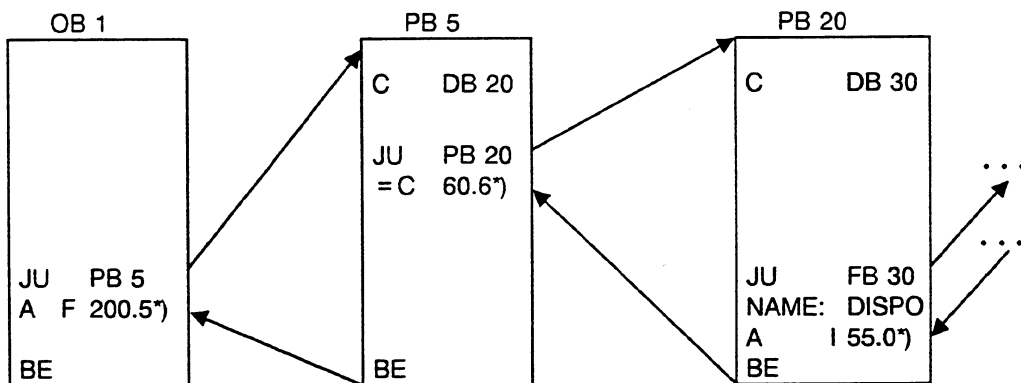
Determine the nesting depth of your program as follows:

- Add all the organization blocks that you have programmed (in example on following page: 4 OBs).
- Add the nesting depth of the individual organization blocks of your STEP 5 program (in example: 2 + 2 + 1 + 0 = 5).
- Total both amounts.  
(In the example shown, the nesting depth is nine 4 + 5 = 9).  
The nesting depth cannot exceed 40.

A **block start address** specifies the vector address to locate any block in the user memory. This is the address in which the first STEP 5 statement of a block is located in the user memory.

The system program enters the start addresses of all programmed blocks in a block address list. The list is in data block DB 0. This enables the CPU to find a block in memory when it is called (e.g., using the JU/JC xx or CDB operations). The system program handles DB 0; you cannot call it.

The CPU stores a return address every time a new block is called. After program processing calls and processes a new block, this return address enables program processing to find the block from which the call originated. The return address is the address of a register in the memory. This register contains the STEP 5 statement that follows the block call statement. The CPU also stores the start address and length of the data block that is current at this location.



\* Indicates a return address

The microprogram enters the following in the **block stack (BSTACK)**: the BSTACK is filled from the bottom. The first entry corresponds to BSTACK element 40, the second to BSTACK element 39, etc.

The BSTACK is filled when it contains 40 nested blocks that are not completely processed (i.e., after the fortieth entry, BSTACK element 1). The CPU goes into the STOP mode if the permissible nesting depth is exceeded.

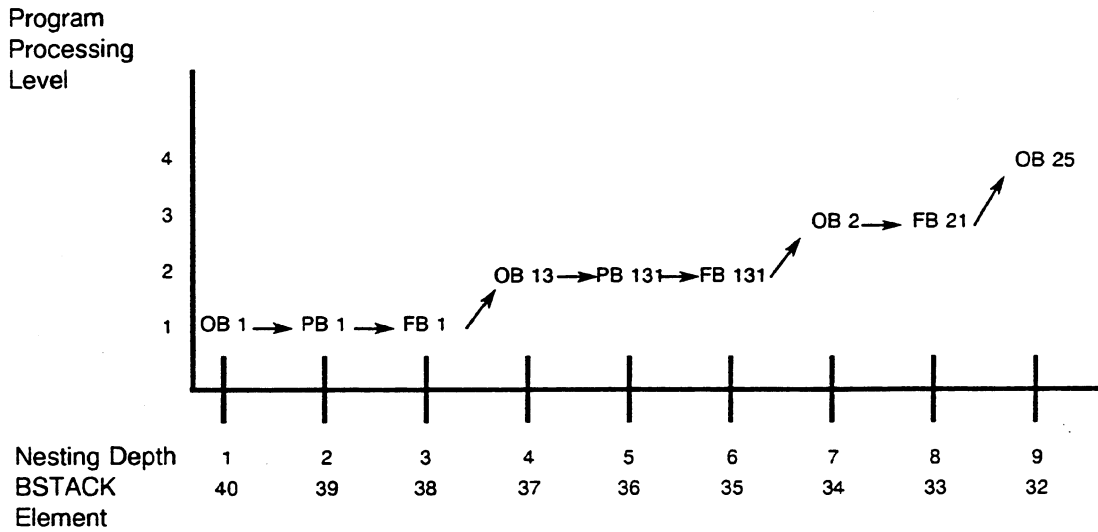


Fig. 3.1 Block Nesting Depth and the Block Stack (BSTACK)

### 3.1.2 Program Storage

You must load your program into the user memory so that the CPU can process it. Use either of the following three methods:

- a) Use the RAM that is integrated into CPU 946/947 and, where needed, plug-in **RAM submodules** with the 355 memory module to transfer your program directly from a programmer to the CPU.

You can change the memory content of a RAM. A central back-up battery in the PLC prevents deletion of your program in the memory if the power goes off. (See the S5-155U Central Controller Housing Hardware and Installation Guide for more information on the back-up battery.)

All programmed blocks are stored in random order in the RAM. If you change a block, the sequence of the blocks in the RAM changes.

If you are using a **RAM module with back-up battery** to store your user program, you can remove it from the CPU without losing any data. A separate battery protects the module against loss of data and makes sure that the data remain available until they are used again.

- b) Store your user program permanently in the 355 module with plug-in EPROM submodules. Your program is completely protected in EPROM submodules even when the power goes off, regardless of back-up battery operation.

Blow EPROM submodules directly on a programmer, for which the mode WORD/FIELD must be selected (see STEP 5 manual). Then plug them into the PLC (355 memory submodule).

- c) You cannot change the contents of an EPROM submodule. For this reason, data blocks that contain variable data and that have to be changed during the course of your program must be in a RAM submodule or in the integrated RAM. You can use special function OB254/OB255 in your STEP 5 program to copy data blocks from the EPROM to the RAM.

When you use a programmer to read data blocks out of the EPROM, change them, and then save them online, the system program stores them in the available RAM.

If the CPU detects an error when checking the user memory during the restart phase, it goes into the STOP mode and requests an overall reset. After performing an overall reset, reload your program into the RAM memory.

An overall reset does not change EPROM submodules.



#### **Caution:**

Buffered RAM modules must not be programmed via the EPROM interface of the PG as they can be destroyed this way.

### 3.1.3 Processing the STEP 5 User Program

The CPU can process the user program in various ways. The most common form of processing for programmable controllers is **cyclic program processing**.

After system start-up, the system program calls organization block OB 1. The CPU begins there with the first STEP 5 statement of the user program and processes all statements sequentially. After processing all the blocks that OB 1 calls, the CPU reaches the end of OB 1. From this point, the system program is called.

The system program does the following:

- updates the process image outputs
- updates the output interprocessor communication flags
- triggers cycle time monitoring
- updates the process image inputs
- updates the input interprocessor communication flags

Then a new cycle with no waiting period starts again by calling OB 1.

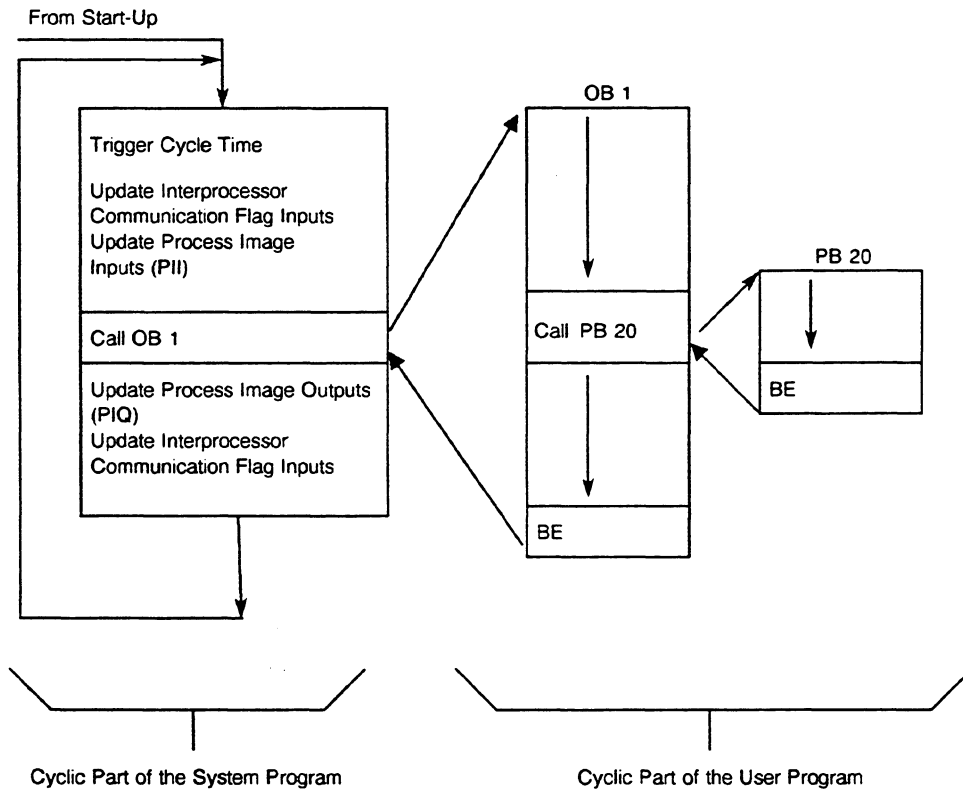


Fig. 3.2 Cyclic Program Processing

### Cycle Time

The system program monitors the time that the CPU needs to process the user program. At the beginning of program processing, the system program starts the cycle time to be monitored.

The default for the cycle monitoring time is 200 ms. You can also set the cycle monitoring time by assigning parameters in extended data block DX 0 or by programming OB 31 (see Chapter 7).

The *total cycle time* consists of the following: scan time of the user program, scan time of the cyclic part of the system program (see Figure 3-3)

The scan time of the user program consists of the total of the scan times of all blocks called during one program pass (from the call of OB 1 to the end of its processing, including all blocks of time and interrupt driven program processing).

Consequently, if you call a specific block "n" number of times, you must multiply its scan time by "n" to arrive at a total.

Note: At the end of each cycle, the actual cycle time needed is stored in a system data word (RS area) where you can read it (see Chapter 8).

### Process Image Input (PII) and Output (PIQ) Tables

Before every STEP 5 program cycle begins, the CPU reads the signal states of the input peripheral modules and transfers them to the process image input table (in the system data memory of the CPU). Using STEP 5 statements, the user program calculates the process image outputs from the process image inputs. After processing the STEP 5 program, the CPU transfers the signal states of the process image outputs to the output peripheral modules.

Therefore, the process image is a memory area whose contents are output to the peripherals or are read in from the peripherals once per cycle. This procedure prevents oscillation of outputs during a cycle with multiple setting and resetting. With multiple switching, the status of the inputs also remains stable. This keeps the load of bus accesses to a minimum on the cycle.

#### NOTE

**A process image exists only for input/output bytes of "P" peripherals with byte addresses from 0 to 127.**

You can use special function OB 126 (transfer process image tables) to define four different process image tables. You can read these into or output them from any program processing level (see section 6.5).

### **Interprocessor Communication Flags**

Interprocessor communication (IPC) flags exchange data between individual CPUs (multiprocessing) or between a CPU and communications processors.

The system program reads in the input IPC flags of the CPU before STEP 5 program processing begins. After the STEP 5 program is processed, the system program transfers the output IPC flags to the coordinator and to the communications processors.

### **Interrupt Points**

Cyclic program processing can be interrupted briefly by the following:

- interrupt-driven program processing (external process interrupts/HW signal interrupts)
- time-driven program processing (internal time interrupts)

Cyclic program processing can also be interrupted or aborted by the following:

- a programmable controller hardware (HW) fault or a program error
- operator intervention (e.g., setting the STOP function with the programmer online or setting the RUN/STOP switch to the STOP position)
- the STEP 5 operation STP

### 3.1.4 Requirements for Program Use

You can influence the CPU during system start-up, during cyclic program processing, and in case of an error in the following two ways:

- a) by programming organization blocks OB 1 to OB 39 (see section 2.2.3)
- b) by programming extended data block DX 0 (see Chapter 7)

#### for a) Programming Organization Blocks

OB 1 to OB 39 are the interfaces between the system program and the user program. On the one hand, the system program calls them, and on the other hand, you load them with a STEP 5 user program like other programmable logic blocks. You can call additional blocks in these organization blocks. You can enter a STEP 5 program into these blocks to specify the reaction of the CPU to certain events.

The system program can call OB 1 through OB 39 as soon as you load them into the program memory. You can also load them when the CPU is in the RUN mode.

#### for b) Programming Extended Data Block DX 0

You can program DX 0 to affect the operation of the CPU.

The functions that the system program carries out are standard defaults. You can change the standard default settings of some system program functions by setting specific parameters in DX 0. Similarly to organization blocks, you can load DX 0 into the program memory while the CPU is running. However, **DX 0 does not take effect until the next cold restart.**

If you do not program DX 0, the defaults apply.



### 3.2 STEP 5 Operations Overview

The STEP 5 operations set is made up of the following three types:

- **Basic operations:** You can program these operations in all programmable logic block types (OBs, PBs, SBs, FBs, and FXs).
- **Supplementary operations:** You can program these operations only in function blocks and extended function blocks (FBs, FXs).
- **System operations:** You can program these operations only in function blocks and extended function blocks (FBs, FXs).

These three types of STEP 5 operations are described in the following sections and in the List of Operations.

Chapter 2 describes operand structure, data types, block types, and block techniques.

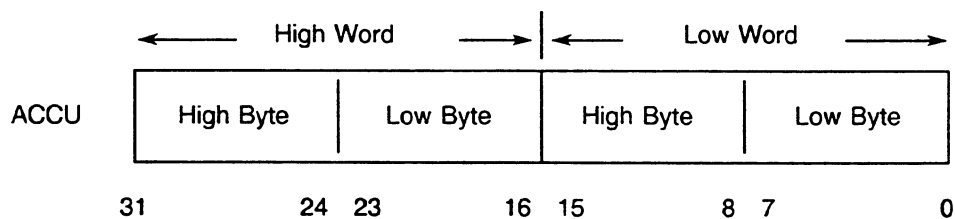
Chapter 9 provides a detailed description of memory access operations using absolute addresses, which apply only in special cases and which should be used only by experienced programmers.

Functionally, the STEP 5 operations can be divided into the following three groups:

- Binary functions – These include Boolean logic operations, set/reset operations, and timer and counter operations.
- Digital functions – These include load and transfer operations, comparison operations, and arithmetic operations.
- Executive functions – These include jump operations, stop operations, block end operations, and block call operations for generating or calling a data block.

#### Accumulators as Working Registers

Most STEP 5 word operations use two 32-bit registers as the source of operands and the destination for results. These two registers are called accumulator 1 (ACCU 1) and accumulator 2 (ACCU 2).



ACCU 3 and ACCU 4 are also available as 32-bit working registers.

The STEP 5 operation to be carried out can affect the accumulators. The working principle of the accumulators is that of stacking. ACCU 1 and ACCU 2 are loaded according to a different principle than ACCU 3 and ACCU 4.

### Examples:

- ACCU 1 is always the destination in load operations. A load operation shifts the old contents of ACCU 1 to ACCU 2 (stack lift) (with exception of LIR and LDI, see Chapter 9). Load operations do not change ACCU 3 and ACCU 4.
- Arithmetic operations combine the contents of ACCU 1 with the contents of ACCU 2 logically and write the result in ACCU 1. They transfer the contents of ACCU 3 to ACCU 2 and the contents of ACCU 4 to ACCU 3 (stack drop).
- When a constant is added to the contents of ACCU 1 (ADD BN/KF/DH), then ACCU 2, ACCU 3, and ACCU 4 are not changed.

### Condition Codes

Some operations process information in single bits and some process information in words (8, 16, or 32 bits).

All operations include instructions for either setting or evaluating condition codes (see the List of Operations). Corresponding to the two groups of operations, there are bit condition codes (bits 0 to 3) and word condition codes (bits 4 to 7). You can display a condition code byte on a programmer via the STATUS function:

| 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

The first bit scan begins a logic combination. At the end of a logic operations sequence (set/reset operations),  $\overline{\text{ERAB}}$  is set to 0. Operations that set  $\overline{\text{ERAB}}$  to 0 (e.g., result assignment "= Q 2.4") reload the result of logic operation (RLO). Reloading the RLO means that the RLO remains constant (see the S5-155U List of Operations). You can evaluate it (e.g., by RLO dependent operations), but you can no longer combine it logically. The RLO is not set up again and  $\overline{\text{ERAB}}$  is not set to 1 until the next logic statement is being processed (i.e., first bit scan).

**RLO** Result of logic operation  
 The RLO is the result of bit-wide logic operations. It is the truth statement for comparison operations (see the S5-155U List of Operations, "Boolean Logic Operations" or "Comparison Operations").

**STA** Status  
 or bit operations, the status bit indicates the logical status of the bit just scanned or set. The status is updated for Boolean logic operations and set/reset operations.  
 Except for: A(  
 O(  
 )  
 O

**OR** Or  
 OR = 1 means that the AND logic operations that follow must be handled before an OR logic operation (AND-before-OR).

**Word Condition Codes**

**OV** (Overflow); the overflow word indicates if the permissible number range was exceeded during a completed arithmetic operation.

**OS** (Stored overflow); the overflow bit is stored. It can be used in several arithmetic operations to indicate if an error occurred because of an overflow (used to recognize arithmetic errors).

**CC 1 and CC 0**  
 The CC 1 and CC 0 word condition codes indicate results of arithmetic, digital logic, comparison, shift, and some conversion operations.

| Word Condition Codes |      | Arithmetic Operations | Digital Logic Operations | Comparison Operations | Shift Operations | With SEE, SED       | Executed Jump Operations |
|----------------------|------|-----------------------|--------------------------|-----------------------|------------------|---------------------|--------------------------|
| 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<br>JN                 |
| 1                    | 0    | Result > 0            | Result ≠ 0               | ACCU 2 > ACCU 1       | Shifted bit = 1  | Semaphore to be set | JP<br>JN                 |
| 1                    | 1    | Division by zero      | —                        | —                     | —                | —                   | —                        |

**Basic operations in all programmable logic block types**

• **Boolean Logic Operations**

| Operation   | Parameter     | Function  |
|-------------|---------------|---|
| )           |               | Close parenthesis (end of a parenthesis operation)                  |
| A (         |               | Combine expression enclosed in parentheses through logic AND.       |
| O (         |               | Combine expression enclosed in parentheses through logic OR.        |
| O           |               | Combine preceding and following AND operations through logic OR.    |
| A .. / O .. |               | Combine through logic AND/OR:                                       |
| I           | 0.0 to 127.7  | Scan an input for signal level 1.                                   |
| Q           | 0.0 to 127.7  | Scan an output for signal level 1.                                  |
| F           | 0.0 to 255.7  | Scan a flag for signal level 1.                                     |
| D           | 0.0 to 255.15 | Scan a bit in the data word of the opened DB/DX for signal level 1. |
| S           | 0.0 to 4095.7 | Scan an S flag for signal level 1                                   |
| N I         | 0.0 to 127.7  | Scan an input for signal level 0.                                   |
| N Q         | 0.0 to 127.7  | Scan an output for signal level 0.                                  |
| N F         | 0.0 to 255.7  | Scan a flag for signal level 0.                                     |
| N D         | 0.0 to 255.15 | Scan a bit in the data word of the opened DB/DX for signal level 0. |
| N S         | 0.0 to 4095.7 | Scan an S flag for signal level 0                                   |
| T           | 0 to 255      | Scan a timer for signal level 1.                                    |
| N T         | 0 to 255      | Scan a timer for signal level 0.                                    |
| C           | 0 to 255      | Scan a counter for signal level 1.                                  |
| N C         | 0 to 255      | Scan a counter for signal level 0.                                  |

The CPU processes logic functions according to Boolean algebra with a bracketing technique. You can enter a logically correct equation directly into a programmer without converting it. Then the CPU processes it directly.

Example:  $(I5.1 \wedge I34.5) \vee (I7.2 \wedge F112.2) = Q4.6$

STEP 5:     :O(  
               :A I5.1  
               :A I34.5  
               :)  
               :O(  
               :A I7.2  
               :A F112.2  
               :)  
               := Q 4.6

The Boolean logic operations generate the RLO as a result. They work like load operations.

At the beginning of a logic operations sequence, the first logic operation (first bit scan) forms the RLO based on the following two factors: the scanned signal status, whether the scanned signal status is negated (N = negation). However, formation of the RLO in this case does not depend on the type of logic operation ("O" = OR, "A" = AND).

In a logic operations sequence, a logic operation forms the RLO based on the following three factors: type of logic operation, previous RLO, scanned signal status. An operation that reloads the RLO (therefore, ERAB = 0, e.g., set/reset operation) terminates a logic operations sequence.

Then, you can evaluate the RLO, but you cannot combine it any further.

**Example:**

| Program | Status | RLO | ERAB   |
|---------|--------|-----|--|
| :       | 0      | 0   | 0 ← RLO reloaded                                       |
| = Q 0.0 | 1      | 1   | 1 ← First bit scan                                     |
| A I 1.0 | 1      | 1   | 1  |
| A I 1.1 | 0      | 0   | 1  |
| A I 1.2 | 0      | 0   | 0 ← RLO reloaded, end of the logic operations sequence |
| = Q 0.1 |        |     |  |

• **Basic Set/Reset Operations for Binary Parameters**

| Operation          | Parameter     | Function                            |
|--------------------|---------------|-------------------------------------|
| S .. / R .. / = .. |               | Set/Reset/Assign:                   |
| 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 data word bit in the opened DB/DX |

• Load, Transfer and Comparison Operations

| Operation   | Parameter                 | Function  |
|-------------|---------------------------|---|
| L .. / T .. |                           | Load/Transfer:  |
| I B         | 0 to 127                  | An input byte from/to the PII   |
| I W         | 0 to 126                  | An input word from/to the PII   |
| I D         | 0 to 124                  | An input double word from/to the PII                                    |
| Q B         | 0 to 127                  | An output byte from/to the PIQ  |
| Q W         | 0 to 126                  | An output word from/to the PIQ  |
| Q D         | 0 to 124                  | An output double word from/to the PIQ                                   |
| F Y         | 0 to 255                  | A flag byte   |
| F W         | 0 to 254                  | A flag word   |
| F D         | 0 to 252                  | A flag double word  |
| D R         | 0 to 255                  | A data byte (right byte) from DB/DX                                     |
| D L         | 0 to 255                  | A data byte (left byte) from DB/DX                                      |
| D W         | 0 to 255                  | A data word from DB/DX  |
| D D         | 0 to 254                  | A data double word from DB/DX   |
| S Y         | 0 to 4095                 | S flag byte   |
| S W         | 0 to 4094                 | S flag word   |
| S D         | 0 to 4092                 | S flag double word  |
| P Y         | 0 to 127                  | A peripheral byte of the digital inputs or outputs ("P" area)           |
| P Y         | 128 to 255                | A peripheral byte of the analog or digital inputs or outputs ("P" area) |
| O Y         | 0 to 255                  | A byte of the extended I/O peripherals area ("O" area)                  |
| P W         | 0 to 126                  | A peripheral word of the digital inputs or outputs ("P" area)           |
| P W         | 128 to 254                | A peripheral word of the analog or digital inputs or outputs ("P" area) |
| O W         | 0 to 254                  | A word of the extended I/O peripherals area ("O" area)                  |
| L           |                           | Load  |
| D H         | 0 to FFFF FFFF            | A double word in hexadecimal code                                       |
| K M         | 16-bit pattern            | A constant as bit pattern   |
| K H         | 0 to FFFF                 | A constant in hexadecimal code  |
| K F         | -32 768 to + 32 767       | A constant as fixed-point number  |
| K Y         | 0 to 255 for each byte    | A constant, two bytes   |
| K B         | 0 to 255                  | A constant, one byte  |
| K S         | 2 alphanumeric characters | A constant, two ASCII characters  |
| K T         | 0.0 to 999.3              | A time value (constant)   |
| K C         | 0 to 999                  | A count value (constant)  |
| K G         | 1)                        | A constant (32 bits) as floating-point number                           |
| T           | 0 to 255                  | A timer   |
| C           | 0 to 255                  | A counter   |
| LC T        | 0 to 255                  | Load (in BCD code): a timer   |
| LC C        | 0 to 255                  | Load (in BCD code): a counter   |

• Load, Transfer and Comparison Operations (Cont.)

| Operation | Parameter | Function                               |
|-----------|-----------|--|
| !=        |           | Compare:<br>For "equal to" ( ! = )     |
| > <       |           | For "not equal to" ( > < )             |
| >         |           | For "greater than" ( > )               |
| > =       |           | For "greater than or equal to" ( > = ) |
| <         |           | For "less than" ( < )                  |
| < =       |           | For "less than or equal to" ( < = ):   |
| F         |           | Two fixed-point numbers (16 bits)      |
| D         |           | Two fixed-point numbers (32 bits)      |
| G         |           | Two floating-point numbers (32 bits)   |

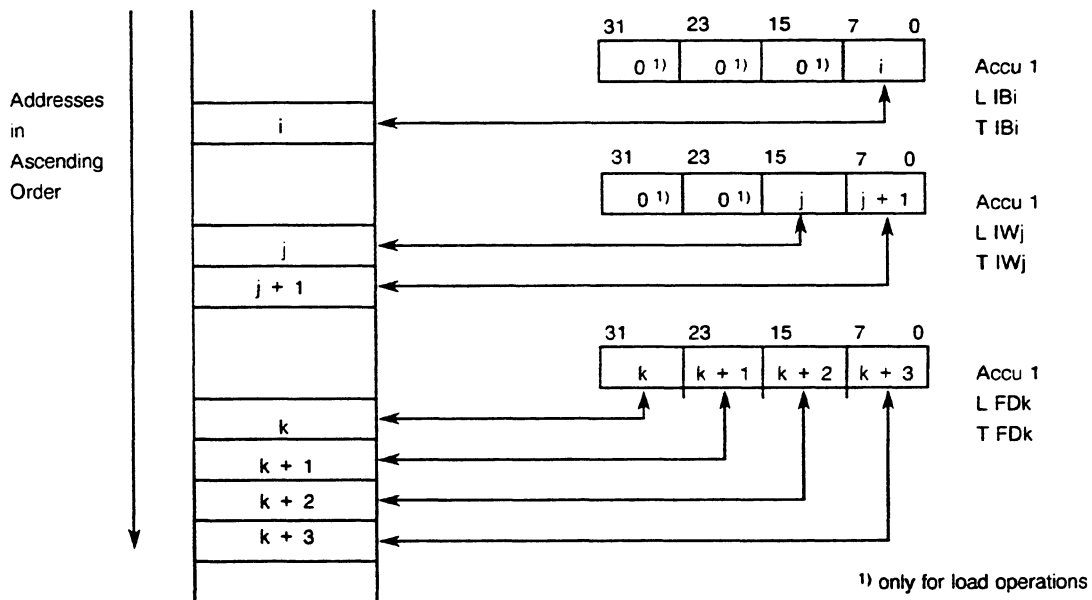
Load operations write the addressed value into ACCU 1. The former contents of ACCU 1 are saved in ACCU 2 (stack lift: exception: LIR and LDI, see Chapter 9).

Transfer operations write the contents of ACCU 1 into the addressed memory register(s).

1)  $\pm 0.1469368 \times 10^{-38}$  to  $\pm 0.1701412 \times 10^{39}$

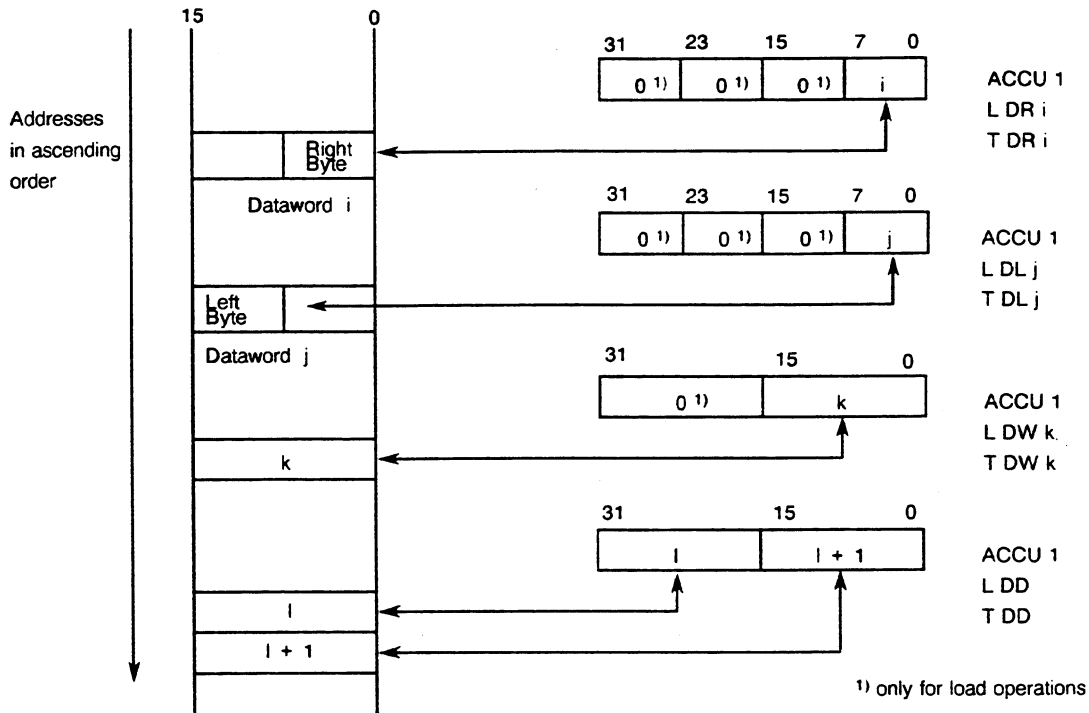
Example: Loading/transferring a byte, word, or double word from/into a memory area organized in bytes (PII, PIQ, flags, I/O peripherals).

- : L IB 5 Load byte 5 of the PII into ACCU 1-LL.
- : L IW 5 Load bytes 5 and 6 of the PII into ACCU 1-L.
- : L FD 10 Load flag bytes 10 to 13 into ACCU 1.



**Example:** Loading/transferring a byte, word, or double word from/into a memory area organized in words.

- : L DR i    Load the right byte from data word i into ACCU 1-LL.
- : L DR j    Load the left byte from data word j into ACCU 1-LL.
- : L DW k    Load the data word k into ACCU 1-L.
- : L DD l    Load the data words l and l + 1 into ACCU 1.



Words or double words are stored in the memory with addresses in ascending order, beginning with the most significant byte or word.

When a byte or word is loaded, the extra bits in ACCU 1 are cleared (set to zero).

Load operations do not affect the condition codes. Transfer operations clear the overflow (OS) bit. Comparison operations generate the RLO and the word condition codes CC1 and CC0. Comparisons are made always between the contents of ACCU 1 and ACCU 2 (see the S5-155U List of Operations and the programming examples).

You can use load and transfer operations to address the I/O periphery as follows:

1. Directly, using the following operations:  
L PY or T PY, L PW or T PW, L OY or T OY, L OW or T OW
2. Indirectly, using the process image and the following operations:  
L IB or T IB, L IW or T IW, L ID or T ID, L QB or T QB, L QW or T QW, L QD or T QD, logic operations

For the transfer operations T PY 0 to T PY 127 and T PW 0 to T PW 126, the PIQ is updated in parallel.



The process image represents a memory area whose contents are output to the periphery (process image output {PIQ} table) or is read in from the periphery (process image input {PII} table) only once per cycle. This prevents any frequent changes in logic levels of a bit in a program cycle from causing the peripheral outputs to oscillate.

Note the following about I/O peripherals:

- A process image input/output table exists for every 128 input/output bytes of the "P" peripherals with byte addresses from 0 to 127.
- No process image table exists for the entire area of the "O" peripherals and for the area of the "P" peripherals with relative byte addresses from 128 to 255. (For information on address space allocation, see section 8.1.2.)
- You can address the "O" peripherals only via the IM 300, IM 301, and IM 304/IM 314 interface modules. Consequently, you can plug in I/O modules with addresses of the "O" peripherals only into expansion units, not into the central controller. Use jumpers on the interface modules to set the address areas.
- In one expansion unit, you can use either the "P" peripherals or the "O" peripherals only.

**NOTE**

**If you use relative addresses of the "O" peripherals in one or more expansion units, you can no longer use these addresses in the central controller. Otherwise, double addressing results. This is not a problem if you are using the "O" peripheral area but have no I/O peripheral modules plugged into the central controller.**

**To avoid double addressing please note the following:**

**If you use an input module in the extended periphery (O area) in an expansion unit, no input module may be present in the CC under the same I/O address. The same applies to output modules. If you use the whole "O" area, no input/output modules may be present in the CC.**

### • Basic Timer and Counter Operations

To execute a timer using a start operation or to execute a counter using a set operation, you must first load the time or count value into ACCU 1 in BCD code.

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.

| Operation | Parameter   | Function                       |
|-----------|-------------|--------------------------------|
| S P T     | 0 to 255 1) | Start a pulse timer.           |
| S E T     | 0 to 255 1) | Start an extended pulse timer. |
| S D T     | 0 to 255 1) | Start an on-delay timer.       |
| S S T     | 0 to 255 1) | Start a stored on-delay timer. |
| S F T     | 0 to 255 1) | Start an off-delay timer.      |
| R T       | 0 to 255 1) | Reset a timer.                 |
| S C       | 0 to 255    | Set a counter.                 |
| R C       | 0 to 255    | Reset a counter.               |
| C U C     | 0 to 255    | Count up.                      |
| C D C     | 0 to 255    | Count down.                    |

1) Attention:

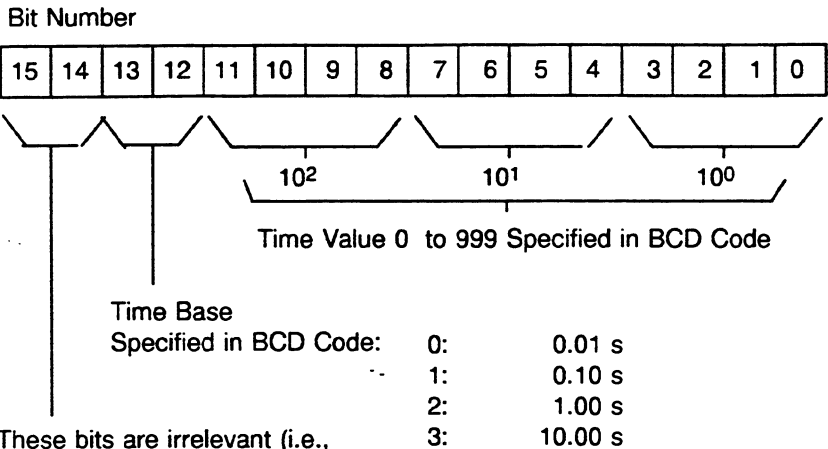
If you made an entry in DX0 indicating the number of timers you want to update, the parameter you use with a timer operation must not exceed that number.

When the SP, SE, SD, SS, and SF timer operations and the S counter operation are executed, the value in ACCU 1 is converted from BCD code into binary form and put into the timer or counter. Then the appropriate operation is executed.

The execution of the operation depends on the RLO and the momentary status of the timer or counter.

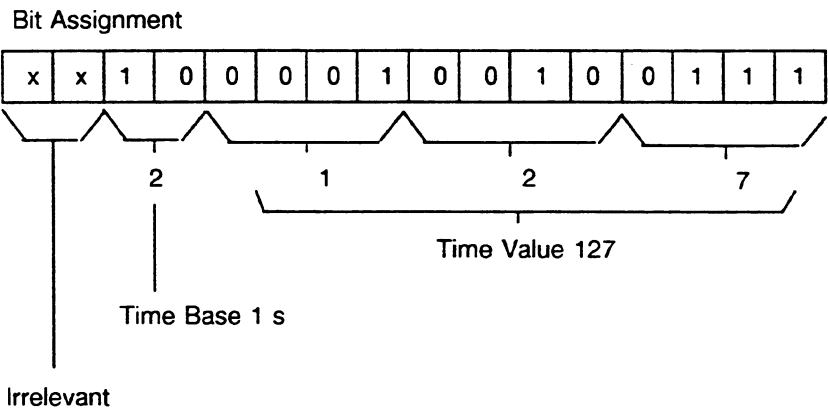
A time value can be loaded indirectly into ACCU 1 if you use the operation L KT. This must have the following structure (the time base is given in the operand for L KT behind the point):

**Word Structure for Time Value**



These bits are irrelevant (i.e., they are ignored when the timer is started).

Example: Setting a Time Value of 127 seconds



**NOTE**

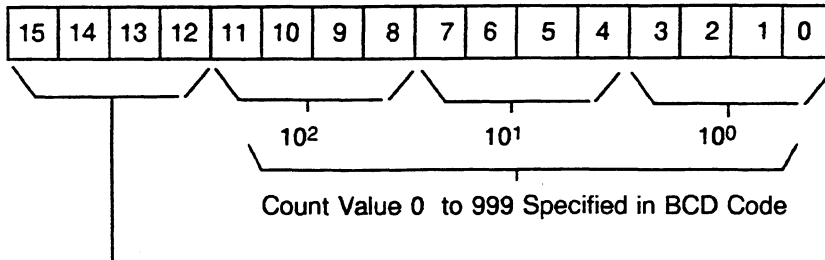
**The start of each timer is related to the tolerance of one time base.**

**If you start a timer with the time base 1 (100 ms) "n" times, you will get an inaccuracy of "n" times 100 ms.**

**Therefore, you should select the smallest possible time base when using timers (time base << time) .**

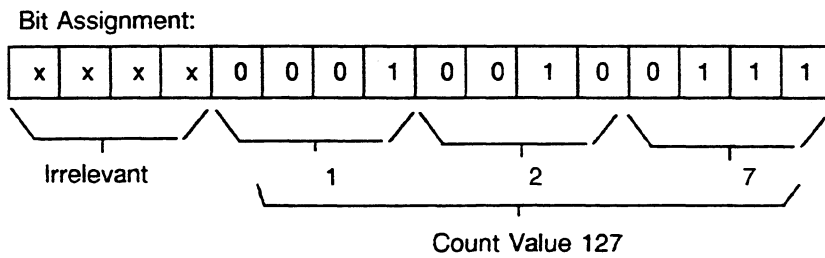
Example: Time Value 4 s    not:    1 s x 4  
    but:    10 ms x 400

**Word Structure for Count Value**



These bits are irrelevant (i.e., they are ignored when the counter is set).

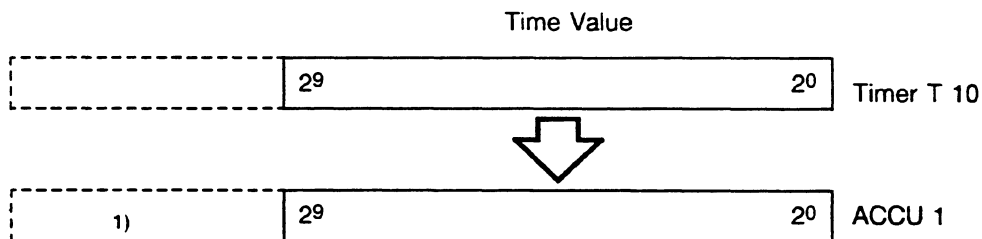
**Example: Setting a Count Value of 127**



In the timer or counter itself, the time or count value is in binary code. If you want to scan the timer or counter, you can load the actual time or count value into ACCU 1 directly or in BCD code.

**Examples**

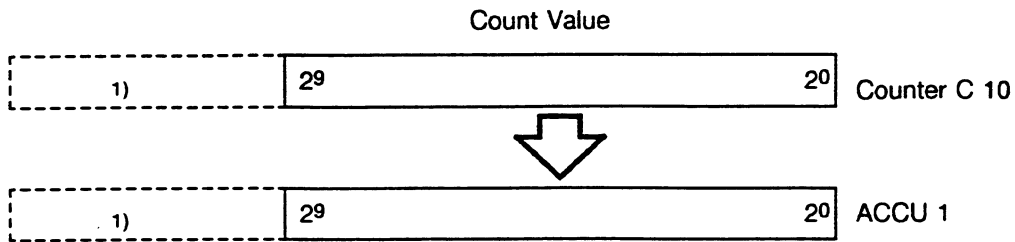
**Direct Loading of Time Value:**



L T 10, direct loading of the binary time of timer T 10 into the accumulator. The time base is not loaded along with the time value.

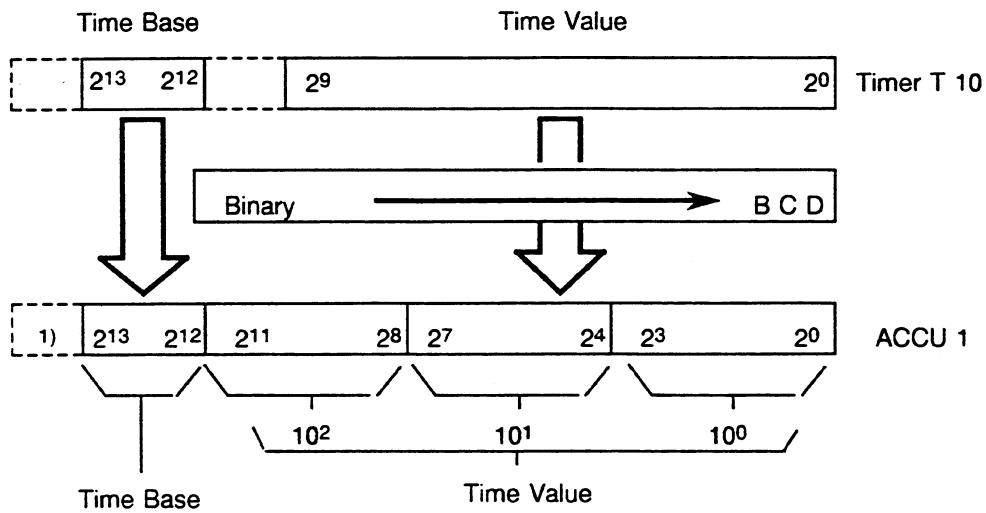
1) The remaining bits are set at "0".

Direct Loading of **Count Value**:



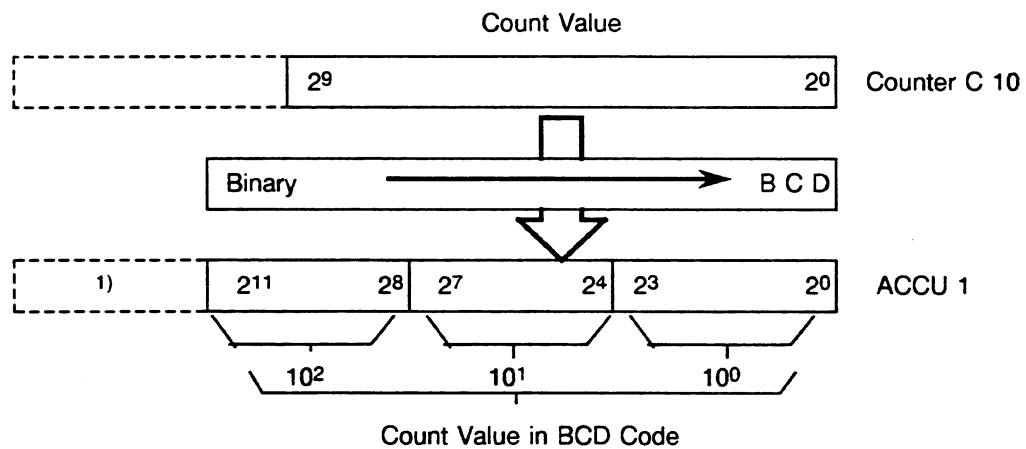
L C 10, direct loading of the Count Value of counter C 10 into the accumulator.

Loading a **Time Value** in BCD Code:



LD T 10, loading the time value and the time base of timer T 10 into the accumulator in BCD code.

Loading a **Count Value** in BCD Code:



LD C 10, loading the count value of counter C 10 into the accumulator in BCD code.

1) The remaining bits are set at 0.

Where loading in BCD code is concerned, status bits 14 and 15 of the timer or 12 to 15 of the counter are not loaded. They are filled with 0 in ACCU 1. Then the value in ACCU 1 can be processed further.

• **Arithmetic Operations**

| Operation  | Parameter | Function   |
|--|-----------|--|
| + F<br>- F<br>x F<br>: F<br>+ G<br>- G<br>x G<br>: G |           | Add/Subtract/Multiply/Divide:<br>Two fixed-point numbers (16 bits)<br><br>Two floating-point numbers (32 bits) |

Arithmetic operations combine the contents of ACCU 1 and ACCU 2 logically (see the S5-155U List of Operations). The result goes to ACCU 1. An arithmetic operation changes the arithmetic registers as follows:

|   |                      |             |
|---|----------------------|-------------|
| Floating-point:                         | Fixed-point:         |             |
| ACCU 1 : = result                       | ACCU 1 L : = result  | (see below) |
| ACCU 2 : = ACCU 3 <--- The old contents | ACCU 2 L : = ACCU 3L |             |
| ACCU 3 : = ACCU 4 of ACCU 2             | ACCU 3 L : = ACCU 4L |             |
| ACCU 4 : = ACCU 4 are lost!             | ACCU 4 : = ACCU 4    |             |

**Division of Two 16-Bit Fixed-Point Numbers**

When the operation for dividing two 16-bit fixed-point numbers is executed, the quotient is in ACCU 1-Low and the remainder is in ACCU 1-High.

Example:   :L KF + 32   Load the first operand into ACCU 2 and  
          :L KF + 10   the second operand into ACCU 1.  
          :F           Divide ACCU 2 (32) by ACCU 1 (10).  
          ::

ACCU assignment:   ACCU 1-High   ACCU 1-Low

|         |         |
|---------|---------|
| 0 0 0 2 | 0 0 0 3 |
|---------|---------|

Remainder   Quotient

Note

The system operations include operations for adding and subtracting double-word, fixed-point numbers.

Section 2.1.4 describes data formats for number representations.

You can load ACCU 3 and ACCU 4 only with the ENT supplementary operation. Use the TAK system operation to swap the contents of ACCU 1 and ACCU 2.

- **Basic Block Call and Block End Operations**

| Operation        |    | Parameter             | Function  |
|------------------|----|-----------------------|---|
| JU<br>JC         |    |                       | Jump unconditionally<br>Jump conditionally (only when RLO = 1):                     |
|                  | OB | 1 to 39 <sup>1)</sup> | To an organization block  |
|                  | OB | 100 to 255            | To call a special OB  |
|                  | PB | 0 to 255              | To a program block  |
|                  | FB | 0 to 255              | To a function block (FB)  |
|                  | SB | 0 to 255              | To a sequence block   |
| DOU              | FX | 0 to 255              | Jump unconditionally to an extended function block (FX)                             |
| DOC              | FX | 0 to 255              | Jump conditionally to an extended function block (FX) (only when RLO = 1):          |
| C                | DB | 2 to 255              | Call a data block (DB).   |
| CX               | DX | 2 to 255              | Call an extended data block (DX).   |
| G                | DB | 2 to 255              | Generate a data block (DB).   |
| GX               | DX | 2 to 255              | Generate an extended data block (DX).   |
| BE<br>BEC<br>BEU |    |                       | Block end<br>Block end, conditional (only when RLO = 1)<br>Block end, unconditional |

<sup>1)</sup> for test purposes only

See section 2.2 for information on block call technique.

### **G DB/GX DX: Generating a Data Block/an Extended Data Block**

The operation G DB x generates a data block with a number x between 2 and 255. Before programming this operation, you must store the number of data words that the new data block is to have in ACCU 1-L (maximum of 4091 DWs). The G DB operation creates the appropriate block header (5 data words) automatically.

The operation GX DX generates an extended data block and works like G DB (permissible parameter is 2 to 255).

The block body of the DB or DX contains random data. For this reason, you must write to a newly generated block first. Only then can usable data be read out.

The system program calls **OB 34** under the following circumstances: if the data block to be generated already exists, if the data block is too long or if the space for the data block in the user memory is insufficient. If OB 34 is not programmed, the CPU goes into the STOP mode.

You can also generate DBs or DXs using special function OB 125 (see Chapter 6).

• "No" Operations

| Operation    | Parameter | Function   |
|--------------|-----------|--|
| NOP0<br>NOP1 |           | No operation (all bits are 0)<br>No operation (all bits are 1) |

• Display Generation Operations

| Operation | Parameter                         | Function  |
|-----------|-----------------------------------|---|
| BLD       | 0 to 255<br><br>130<br>131<br>255 | Automatic input of statement for the programmer: The CPU handles this operation like a "No" operation.<br>Generate a blank line using carriage return.<br>Switch to statement list (STL).<br>Terminate segment. |

• STOP Operation

| Operation | Parameter | Function   |
|-----------|-----------|--|
| STP       |           | The CPU goes into the smooth STOP mode at the end of the cycle or the end of OB1 via the system program. |

Note:

As the STP operation only becomes effective at the end of the cycle, an ISTACK entry is not made. Therefore it is quite difficult to find the cause of the stop condition afterwards via diagnostic aids. Thus, to make diagnostics easier, you should set an identifier before calling the STP operation, e.g. bit pattern in an error DB.



Programming Examples for Logic, Set/Reset, Timer, Counter, and Comparison Operations

• Boolean Logic Operations

AND Operation

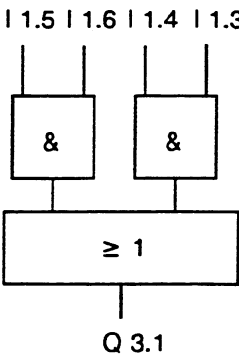
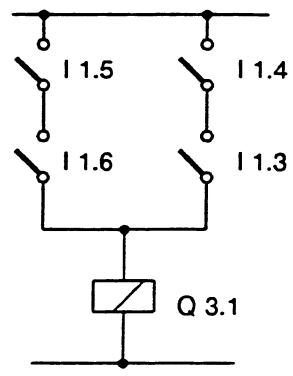
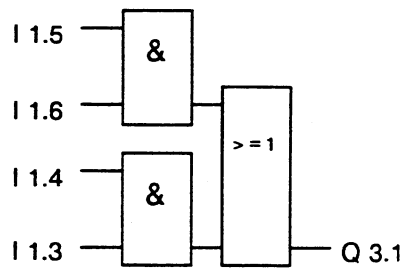
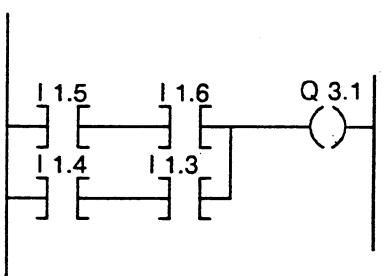
| Example   | Logical Diagram | Circuit Diagram |
|---|-----------------|-----------------|
| <p>Output "Q 3.5" is 1 when all three inputs are 1. The output is 0 if at least one input is 0.</p> <p>The number of scans and the sequence of the logic statements are optional.</p> |                 |                 |
| STL   | CSF             | LAD             |
| <pre> A   I 1.1 A   I 1.3 A   I 1.7 = Q 3.5                     </pre>  |                 |                 |

OR Operation

| Example   | Logical Diagram | Circuit Diagram |
|---|-----------------|-----------------|
| <p>Output "Q 3.2" is 1 when at least one of the inputs is 1. Output "Q 3.2" is 0 when all inputs are 0 simultaneously.</p> <p>The number of scans and the sequence of their programming are optional.</p> |                 |                 |
| STL   | CSF             | LAD             |
| <pre> O   I 1.2 O   I 1.7 O   I 1.5 = Q 3.2                     </pre>  |                 |                 |

• Boolean Logic Operations (Cont.)

AND-before-OR Operation

| Example   | Logical Diagram  | Circuit Diagram   |
|---|--|---|
| <p>Output "Q 3.1" is 1 when at least one AND condition is satisfied. Output "Q 3.1" is 0 when neither of the two AND conditions is satisfied.</p> |   |   |
| STL   | CSF  | LAD   |
| <pre> A   I   1.5 A   I   1.6 O A   I   1.4 A   I   1.3 =   Q   3.1         </pre>  |  |  |

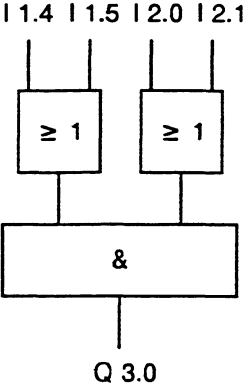
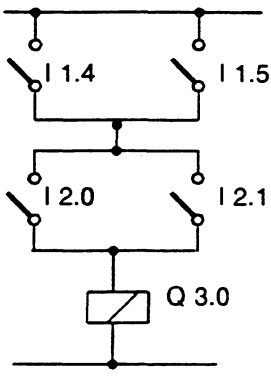
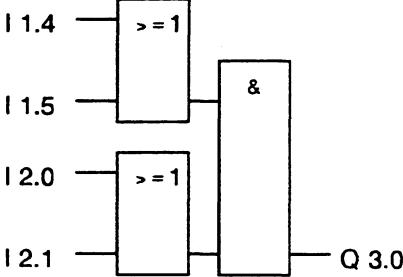
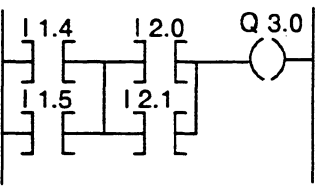
• Boolean Logic Operations (Cont.)

OR-before-AND Operation

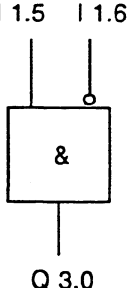
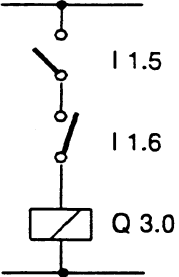
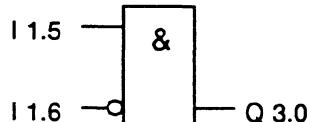
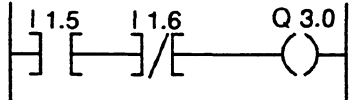
| Example  | Logical Diagram | Circuit Diagram |
|--|-----------------|-----------------|
| <p>Output "Q 2.1" is 1 when one of the following conditions is satisfied:</p> <ul style="list-style-type: none"> <li>• Input "I 6.0" is 1.</li> <li>• Input "I 6.1" and either input "I 6.2" or "I 6.3" are 1.</li> </ul> <p>Output "Q 2.1" is 0 when both of the following conditions are satisfied:</p> <ul style="list-style-type: none"> <li>• Input "I 6.0" is 0.</li> <li>• The AND condition is not satisfied.</li> </ul> |                 |                 |
| STL  | CSF             | LAD             |
| <pre> O   I   6.0 O   I   6.1 A(  I   6.2 O   I   6.3 ) =   Q   2.1     </pre>   |                 |                 |

• Boolean Logic Operations (Cont.)

OR-before-AND Operation

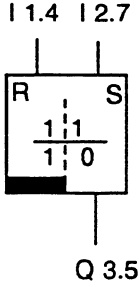
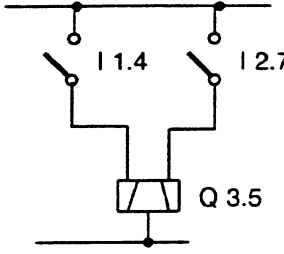
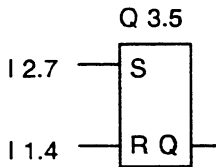
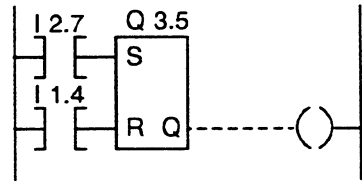
| Example  | Logical Diagram  | Circuit Diagram   |
|--|--|---|
| <p>Output "Q 3.0" is 1 when both OR conditions are satisfied. Output "Q 3.0" is 0 when at least one OR condition is not satisfied.</p> |   |  |
| STL  | CSF  | LAD   |
| <pre> A( O  I 1.4 O  I 1.5 ) A( O  I 2.0 O  I 2.1 ) =  Q 3.0     </pre>  |  |  |

Scan for Signal State 0

| Example   | Logical Diagram   | Circuit Diagram  |
|---|---|--|
| <p>Output "Q 3.0" is 1 only when input "I 1.5" is 1 (normally open contact activated) and input "I 1.6" is 0 (normally closed contact activated).</p> |  |  |
| STL   | CSF   | LAD  |
| <pre> A  I 1.5 AN I 1.6 =  Q 3.0     </pre>   |  |  |

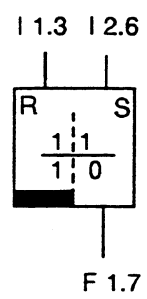
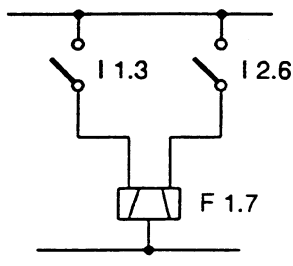
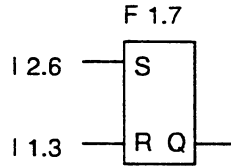
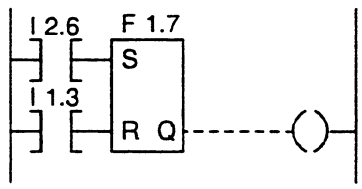
• Set/Reset Operations

RS Flip-Flop for a Latching Signal Output

| Example   | Logical Diagram   | Circuit Diagram  |
|---|---|--|
| <p>A 1 at input "I 2.7" sets flip-flop "Q 3.5" (signal state 1). If the signal state at input "I 2.7" changes to 0, the state of output "Q 3.5" is maintained (i.e., the signal is latched).</p> <p>A 1 at input "I 1.4" resets the flip-flop ("Q 3.5" is signal state 0). If the signal state at input "I 1.4" changes to 0, the state of "Q 3.5" is maintained (i.e., the signal is unlatched). When the SET signal (input "I 2.7") and the RESET signal (input "I 1.4") are applied at the same time, the scanning operation that was programmed last (in this case "A I 1.4") is in effect for the rest of the program. In this example, resetting output "Q 3.5" has priority.</p> |    |   |
| STL   | CSF   | LAD  |
| <pre> A I 2.7 S Q 3.5 A I 1.4 R Q 3.5                     </pre>  |  |  |

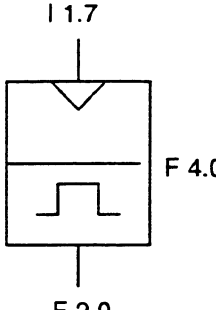
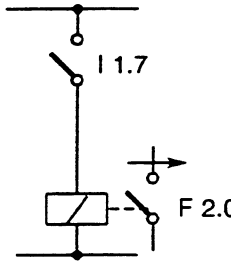
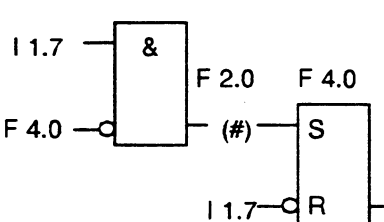
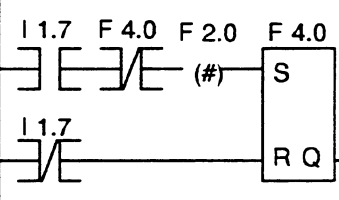
● Set/Reset Operations (Cont.)

RS Flip-Flop with Flags

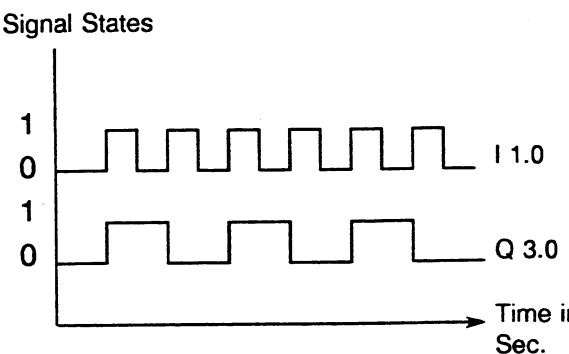
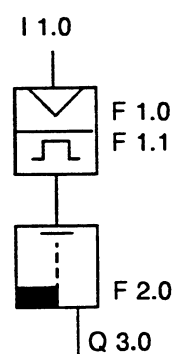
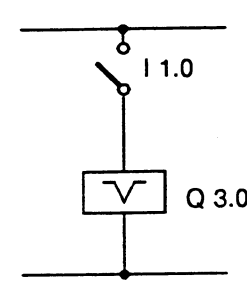
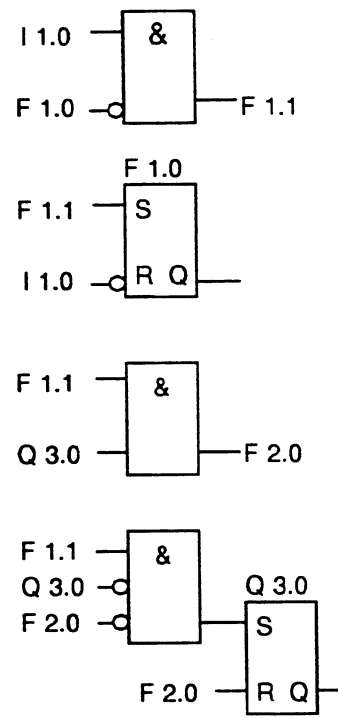
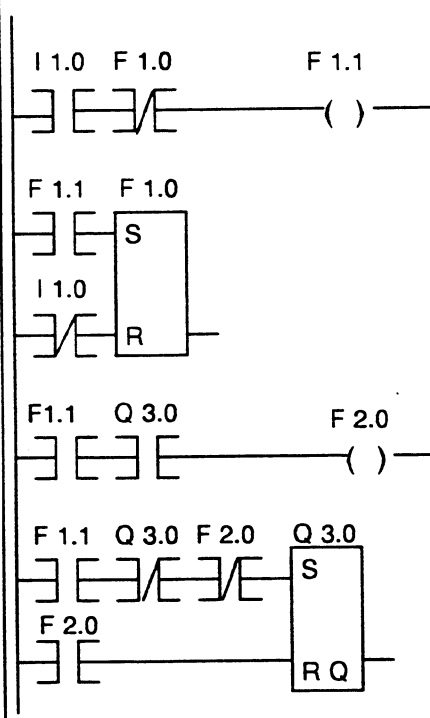
| Example  | Logical Diagram   | Circuit Diagram  |
|--|---|--|
| <p>A 1 at input "I2.6" sets flip-flop "F1.7" (signal state 1). If the signal state at input "I2.6" changes to 0, the state of flag "F1.7" is maintained (i.e., the signal is latched).</p> <p>A 1 at input "I1.3" resets the flip-flop (signal state 0). If the signal state at input "I1.3" changes to 0, flag "F1.7" maintains signal state 0. The signal state of the flag is scanned and transferred to output "Q3.4".</p> <p>When the SET signal (input "I2.6") and the RESET signal (input "I1.3") are applied at the same time, the scanning operation that was programmed last (in this case, "A I1.3") is in effect for the rest of the program. In this example, resetting has priority.</p> |    |    |
| STL  | CSF   | LAD  |
| <pre> A I 2.6 S F 1.7 A I 1.3 R F 1.7                     </pre>   |  |  |

• Set/Reset Operations (Cont.)

Simulating an interval time-delay relay

| Example  | Logical Diagram   | Circuit Diagram  |
|--|---|--|
| <p>At every leading edge of input "I 1.7" the AND condition is satisfied (A I 1.7 and AN F 4.0) and with RLO = "1" the flags F 4.0 ("edge flag") and F 2.0 ("pulse flag") are set.</p> <p>During the next cycle the AND operation A I 1.7 and AN F 4.0 is not satisfied, because the flag F 4.0 was set.</p> <p>The flag F 2.0 is reset.</p> <p>The flag F 2.0 leads to signal state "1" after one complete program cycle.</p> |    |   |
| STL  | CSF   | LAD  |
| <pre> 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                     </pre>   |  |  |

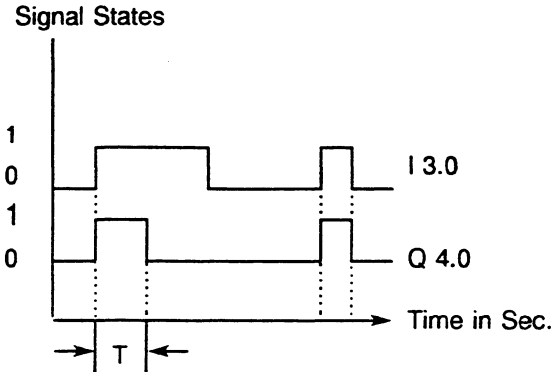
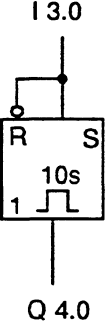
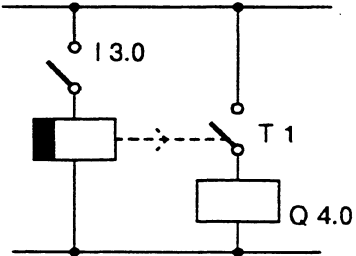
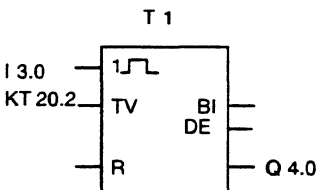
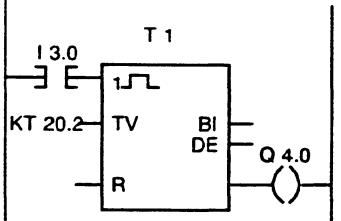
• Set/Reset Operations (Cont.)

| Example  | Timing Diagram   |   |
|--|--|---|
| <p>The binary scaler (output "Q 3.0") changes its state each time "I 1.0" changes its signal state from 0 to 1 (leading edge). Therefore, half the input frequency appears at the output of the memory cell.</p>                 |                                |   |
|  | <p><b>Logical Diagram</b></p>  | <p><b>Circuit Diagram</b></p>  |
| <p><b>STL</b></p>  | <p><b>CSF</b></p>  | <p><b>LAD</b></p>   |
| <pre> 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   Q   3.0 =   F   2.0 A   F   1.1 AN  Q   3.0 AN  F   2.0 S   Q   3.0 A   F   2.0 R   Q   3.0                     </pre> |                               |                               |



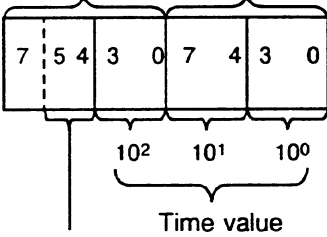
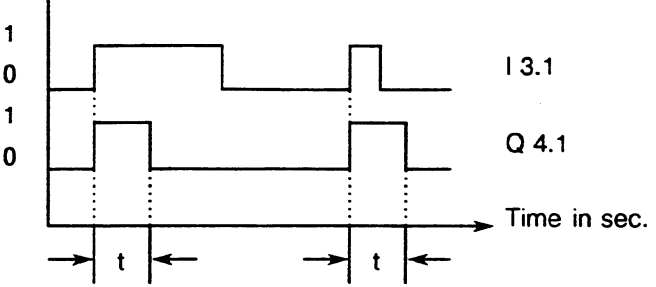
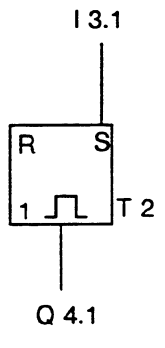
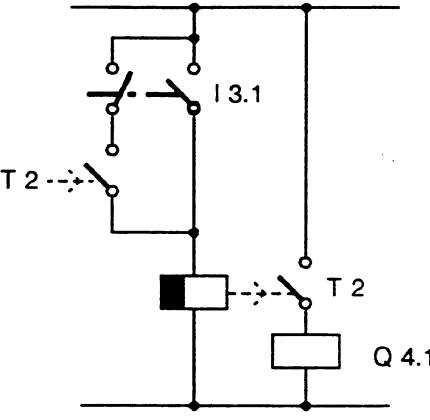
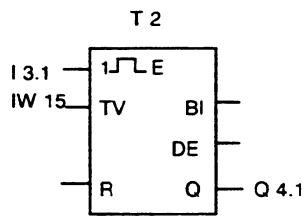
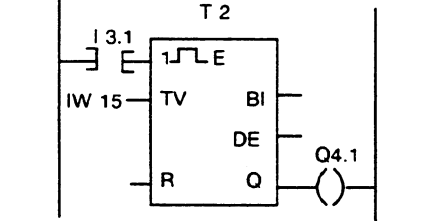
• Timer Operations

Pulse Timer

| Example  | Timing Diagram  |  |
|--|---|--|
| <p>The timer is started during the initial scan when the RLO is 1. Subsequent scans with an RLO of 1 do not affect the timer.</p> <p>When the RLO is 0, the timer is reset (cleared).</p> <p>The scan AT or OT produces the signal 1 as long as the timer is running.</p> <p>KT 10.2</p> <p>The timer is loaded with the given value (10).</p> <p>The number to the right of the decimal point indicates the time base:</p> <p>0 = 0.01 sec. 2 = 1 sec.<br/>1 = 0.1 sec. 3 = 10 sec.</p> <p>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.</p> |   |  |
|  | Logical Diagram   | Circuit Diagram  |
|  |   |   |
| STL  | CSF   | LAD  |
| <p>A I 3.0<br/>L KT 20.2<br/>SP T 1<br/>A T 1<br/>= Q 4.0</p>  |  |  |

• Timer Operations (Cont.)

Extended Pulse Timer

| Example   | Timing Diagram  |  |
|---|---|--|
| <p>The timer is started during the initial scan when the RLO is 1.</p> <p>An RLO of 0 does not affect the timer.</p> <p>The scan AT or OT produces the signal 1 as long as the timer is running.</p> <p>In input word IW15, set the timer with the time of the operand I, Q, F, or D in BCD code (see example IW 15).</p>  <p>Time base</p> | <p>Signal States</p>  <p>Time in sec.</p> |  |
|   | Logical Diagram   | Circuit Diagram  |
|   |   |   |
| STL   | CSF   | LAD  |
| <pre> A   I   3.1 L   IW  15 SE  T   2 A   T   2 =   Q   4.1         </pre>   |    |  |

• **Timer Operations (Cont.)**

On-Delay Timer

| Example   | Timing Diagram   |                               |
|---|--|-------------------------------|
| <p>The timer is started during the initial scan when the RLO is 1. An RLO of 1 during subsequent scans does not affect the timer.</p> <p>When the RLO is 0, the timer is reset (cleared).</p> <p>The scan A T or O T produces the signal 1 when the timer has run out and the RLO is still pending at the input.</p> <p>Load the timer with the indicated time (KT 9.2 = 9 sec.).</p> <p>The number to the right of the decimal point indicates the time base:</p> <p>0 = 0.01 sec. 2 = 1 sec.<br/>1 = 0.1 sec. 3 = 10 sec.</p> | <p>Signal States</p> <p>I 3.5</p> <p>Q 4.2</p> <p>Time in sec.</p> |                               |
|   | <p><b>Logical Diagram</b></p>                                      | <p><b>Circuit Diagram</b></p> |
| <p><b>STL</b></p>   | <p><b>CSF</b></p>  | <p><b>LAD</b></p>             |
| <p>A I 3.5</p> <p>L KT 9.2</p> <p>SD T 3</p> <p>A T 3</p> <p>= Q 4.2</p>  |  |                               |

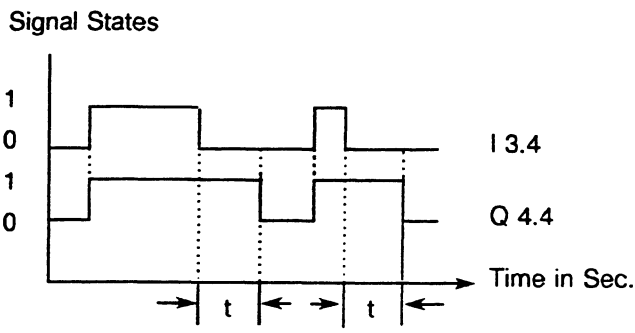
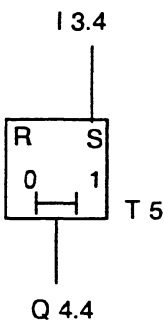
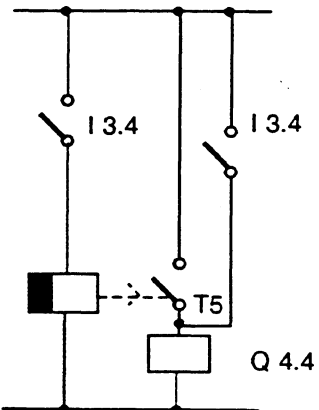
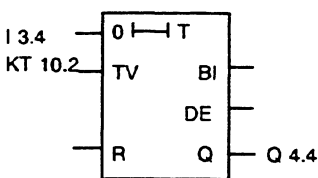
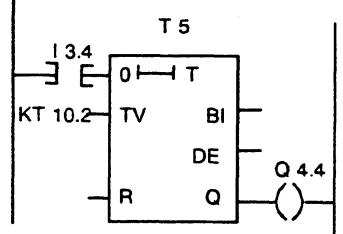
• Timer Operations (Cont.)

Stored On-Delay Timer

| Example  | Timing Diagram                |                               |
|--|-------------------------------|-------------------------------|
| <p>The timer is started during the initial scan when the RLO is 1.</p> <p>An RLO of 0 does not affect the timer.</p> <p>The scan AT or OT produces the signal 1 when the timer has run out. The signal state does not change to 0 until the RT operation resets the timer.</p> |                               |                               |
|  | <p><b>Logical Diagram</b></p> | <p><b>Circuit Diagram</b></p> |
| <p><b>STL</b></p>  | <p><b>CSF</b></p>             | <p><b>LAD</b></p>             |
| <pre> A I 3.3 L KT 20.2 SS T 4 A I 3.2 R T 4 A T 4 = Q 4.3                     </pre>  |                               |                               |

● Timer Operations (Cont.)

Off-Delay Timer

| Example  | Timing Diagram  |  |
|--|---|--|
| <p>When the RLO at the start input changes from 1 to 0, the timer is started. It runs for the length of time programmed.</p> <p>When the RLO is 1, the timer is reset (cleared).</p> <p>The scan AT or OT produces signal state 1 if the timer is running <u>or</u> the RLO at the input is 1.</p> |   |  |
|  | Logical Diagram   | Circuit Diagram  |
|  |   |   |
| STL  | CSF   | LAD  |
| <p>A I 3.4<br/>L KT 10.2<br/>SF T 5<br/>A T 5<br/>= Q 4.4</p>  |  |  |

• Counter Operations

Setting a Counter

| Example   |     | Circuit |  |
|---|-----|---------|--|
| <p>The counter is started during the initial scan when the RLO is 1. During additional scans, the counter is not affected (regardless of whether the RLO is 1 or 0). A renewed initial scan with RLO equal to 1 sets the counter again (edge evaluation).</p> <p>The flag necessary for edge evaluation of the set input is incorporated in the counter word.</p> <p>BI and DE are digital outputs of the counter register. The count at output BI is in binary code. The count at DE is in BCD code.</p> |     |         |  |
| STL   | CSF | LAD     |  |
| <p>A I 4.1<br/>L IW 20<br/>S C 1</p>  |     |         |  |

• Counter Operations (Cont.)

Resetting a Counter

| Example  |     | Circuit |  |
|--|-----|---------|--|
| <p>An RLO of 1 resets the counter.</p> <p>An RLO of 0 does not affect the counter.</p> |     |         |  |
| STL  | CSF | LAD     |  |
| <pre> A   I   4.2 R   C   1 A   C   1 =   Q   2.4         </pre>                       |     |         |  |

• Counter Operations (Cont.)

Counting Up

| Example  |     | Circuit |
|--|-----|---------|
| <p>The value of the referenced counter is increased by 1 to a maximum count of 999. Only a leading edge change (from 0 to 1) of the logic operation programmed before CU triggers the CU function. The flags necessary for edge evaluation of the counter inputs are incorporated in the counter word.</p> <p>You can have a counter with two different inputs as an up and down counter by using the two separate edge flags for CU and CD.</p> |     |         |
| STL  | CSF | LAD     |
| <pre> A   I   4.1 CU  C   1         </pre>   |     |         |



• Counter Operations (Cont.)

Counting Down

| Example   |     | Circuit |
|---|-----|---------|
| <p>The value of the referenced counter is decreased by 1, to a minimum of 0. Only a leading edge change (from 0 to 1) of the logic operation programmed before CD triggers the CD function. The flags necessary for edge evaluation of the counter inputs are incorporated in the counter word.</p> <p>You can have a counter with two different inputs as an up and down counter by using the two separate edge flags for CU and CD.</p> |     |         |
| STL   | CSF | LAD     |
| <pre> A      I      4.0 CD    C      1         </pre>   |     |         |

• Comparison Operations

Compare for "Equal to"

| Example   |      | Diagram |  |  |
|---|------|---------|--|--|
| <p>The first operand is compared to the second operand by the comparison operation. The RLO of the comparison is binary.</p> <p>RLO = 1 - Comparison is satisfied if ACCU 1-Low = ACCU 2-Low.<br/>                     RLO = 0 - Comparison is not satisfied if ACCU 1-Low ≠ ACCU 2-Low.</p> <p>The condition codes CC 1 and CC 0 are set as explained in Chapter 3.</p> <p>ACCU 2-High and ACCU 1-High are not involved in the operation for a 16-bit fixed-point comparison.</p> <p>For fixed-point comparison (! = D {double word}) and floating-point comparison (! = G), the entire contents of ACCU 1 and ACCU 2 (32 bits) are compared to each other. During comparison, the number representation of the operands is considered (i.e., the contents of ACCU 1-Low and ACCU 2-Low is interpreted here as a fixed-point number {word}).</p> |      |         |  |  |
| <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="width: 20px; text-align: center;">0</td><td style="width: 40px; text-align: center;">IB19</td></tr> </table> ACCU 2-Low  | 0    | IB19    |  |  |
| 0   | IB19 |         |  |  |
| <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td style="width: 20px; text-align: center;">0</td><td style="width: 40px; text-align: center;">IB20</td></tr> </table> ACCU 1-Low  | 0    | IB20    |  |  |
| 0   | IB20 |         |  |  |
| STL   | LAD  | CSF     |  |  |
| <pre>                     L   IB   19                     L   IB   20                     !=F                     =   Q    3.0                 </pre>   |      |         |  |  |

• Comparison Operations (Cont.)

Compare for "Not Equal to"

| Example   |   | Circuit    |            |  |  |
|---|---|------------|------------|--|--|
| <p>The first operand is compared to the second operand by the comparison operation. The RLO of the comparison is binary.<br/>                     RLO = 1 - Comparison is satisfied if ACCU 1-Low <math>\neq</math> ACCU 2-Low.<br/>                     RLO = 0 - Comparison is not satisfied if ACCU 1-Low = ACCU 2-Low.</p> <p>The condition codes CC 1 and CC 0 are set as explained in Chapter 3.</p> <p>ACCU 2-High and ACCU 1-High are not involved in the operation for a 16-bit fixed-point comparison.<br/>                     ACCU 2-High and ACCU 1-High are involved in the comparison for fixed-point (32 bits) and floating-point comparisons.<br/>                     This same information applies to the comparison operations for "greater than," "greater than or equal to," "less than," and "less than or equal to" (see the S5-155U List of Operations).<br/>                     During comparison, the number representation of the operands is considered (i.e., the contents of ACCU 1-Low and ACCU 2-Low are interpreted here as a fixed-point number).</p> |   |            |            |  |  |
| <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">0</td> <td style="text-align: center;">IB 21</td> <td>ACCU 2-Low</td> </tr> </table>   | 0   | IB 21      | ACCU 2-Low |  |  |
| 0   | IB 21   | ACCU 2-Low |            |  |  |
|   | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">DW 3</td> <td>ACCU 1-Low</td> </tr> </table> | DW 3       | ACCU 1-Low |  |  |
| DW 3  | ACCU 1-Low  |            |            |  |  |
| STL   | LAD   | CSF        |            |  |  |
| <pre> L   IB  21 L   DW   3 &gt; &lt; F =   Q   3.1                     </pre>  |   |            |            |  |  |

### 3.3 Supplementary Operations

You can use supplementary operations only in function blocks (FBs) and extended function blocks (FXs). The system operations are similar to the supplementary operations.

You can use the system operations to overwrite the memory at optional locations or to change the contents of the working registers of the CPU. *Only experienced computer programmers with good knowledge of the CPU 946/947 should use the system operations and then only with extreme caution.*

See Chapter 9 for more information on memory access via absolute addressing.

You can write operations in function blocks only in the STL method of representation. You cannot program function blocks in graphic form (LAD and CSF methods of representation).

There follows a description of the additional operations which can only be used in function blocks.

This section describes the supplementary operations and covers possible combinations of substitution operations with actual operands.

- **Supplementary Boolean Logic Operations of Formal Operands**

| Operation | Parameter            | Function  |
|-----------|----------------------|---|
| A =       | <input type="text"/> | AND function, scan a formal operand for 1   |
| AN =      | <input type="text"/> | AND function, scan a formal operand for 1   |
| O =       | <input type="text"/> | AND function, scan a formal operand for 0   |
| ON =      | <input type="text"/> | OR function, scan a formal operand for 1  |
|           | ↑                    | Set formal operand (description of the block parameter)   |
|           |                      | Permissible actual operands include inputs, outputs, data and flags addressed in binary form (parameter types I and Q, data type BI), as well as timers and counters (parameter types T and C). |

See section 2.3.2 for information on programming with formal operands.

• Bit Test Operations

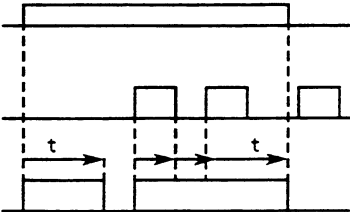
| Operation       | Parameter     | Function   |
|-----------------|---------------|--|
| TB...<br>TBN... |               | Scan for signal state 1 of:<br>Scan for signal state 0 of: |
| I               | 0.0 to 127.7  | Input bit (PII)  |
| Q               | 0.0 to 127.7  | Output bit (PIQ)   |
| F               | 0.0 to 255.7  | Flag bit   |
| D               | 0.0 to 255.15 | Data word bit  |
| T               | 0.0 to 255.15 | Bit value in timer   |
| C               | 0.0 to 255.15 | Bit value in counter                                       |
| RI              | 0.0 to 255.15 | Bit in the RI area   |
| RJ              | 0.0 to 255.15 | Bit in the RJ area   |
| RS              | 0.0 to 255.15 | Bit in the RS area   |
| RT              | 0.0 to 255.15 | Bit in the RT area   |

The bit test operations scan the status of a bit and indicate it via the RLO.

• Supplementary Set/Reset Operations

| Operation  | Parameter  | Function  |
|--|--|---|
| S = <input type="text"/>                                 |  | Set a formal operand (binary).<br>(parameter type I or Q, data type BI)   |
| RB = <input type="text"/>                                |  | Reset a formal operand (binary).<br>(parameter type I or Q, data type BI)   |
| RD = <input type="text"/>                                |  | Reset a formal operand for timers and<br>counters (digital).<br>(parameter type T or C)   |
| = = <input type="text"/><br>↑<br>Formal<br>operand       |  | Assign the value of the RLO to the formal<br>operand. (parameter type I or Q, data type BI)   |
| SU...<br>RU...<br>I<br>Q<br>F<br>D<br>T<br>C<br>RQ<br>RJ | 0.0 to 127.7<br>0.0 to 127.7<br>0.0 to 255.7<br>0.0 to 255.15<br>0.0 to 255.15<br>0.0 to 255.15<br>0.0 to 255.15<br>0.0 to 255.15<br>0.0 to 255.15 | Set unconditionally<br>Reset unconditionally:<br><br>Input bit<br>Output bit<br>Flag bit<br>Data word bit<br>Bit value in timer<br>Bit value in counter<br>Bit in the RI area<br>Bit in the RJ area |

• Supplementary Timer and Counter Operations

| Operation  | Parameter  | Function  |
|--|------------|---|
| FR   | T 0 to 255 | <p>Enable a timer for cold restart.<br/>This operation is executed only on the leading edge of the RLO (change from 0 to 1). It restarts the timer if the RLO is 1 at the time of the start operation.</p>    |
| FR = <input type="text"/><br><br>RD = <input type="text"/><br><br>SP = <input type="text"/><br><br>SD = <input type="text"/><br><br>SEC = <input type="text"/><br><br>SSU = <input type="text"/><br><br>SFD = <input type="text"/> | C 0 to 255 | <p>Enable a counter.<br/>This operation is executed only on the leading edge of the RLO (change from 0 to 1). It sets the counter or causes it to count up or down if the RLO is 1 at the corresponding operation.</p> <p>Enable a formal operand for cold restart.<br/>(For a description, see FR T or FR C, depending on the formal operand.) (parameter type T or C)</p> <p>Reset a formal operand (digital). (parameter type T or C)</p> <p>Start a Pulse timer specified as a formal operand using the value stored in the accumulator. (parameter type T)</p> <p>Start an On-Delay timer specified as a formal operand using the value stored in the accumulator. (parameter type T)</p> <p>Start an Extended Pulse timer specified as a formal operand using the value stored in the accumulator, or set a counter specified as a formal operand using the count specified in the accumulator. (parameter type T or C)</p> <p>Start a Stored On-Delay timer specified as a formal operand using the value stored in the accumulator, or start the count up of a counter specified as a formal operand. (parameter type T or C)</p> <p>Start an Off-Delay timer specified as a formal operand using the value stored in the accumulator, or start the count down of a counter specified as a formal operand. (parameter type T or C)</p> <p>Formal operand</p> <p>Actual operands permitted include timers and counters. However, SP and SD do not apply to counters. You can specify the time or count value as you do with the basic operations or as a formal operand as follows:<br/>Set the time or count value with the value of the IW, QW, FW, or DW (parameter type I, data type W) specified as a formal operand in BCD code or as a constant (parameter type D, data type KT or KC).</p> |

Examples

| Function Block Call   | Program in the Function Block  | Program  |
|---|--|--|
| <b>:JU FB 203</b><br>NAME : <b>EXAMPLE1</b><br>STRT : I 10.3<br>TIMR : T 17<br>OUT1 : Q 18.4                                  | :A = STRT<br>:L KT 010.2<br>:SSV = TIMR<br>:U = TIMR<br>:= = OUT1  | :A E 10.3<br>:L KT 010.2<br>:SS T 17<br>:A T 17<br>:= Q 18.4   |
| <b>:JU FB 204</b><br>NAME : <b>EXAMPLE2</b><br>USTR : I 10.5<br>DSTR : I 10.6<br>STRT : I 10.7<br>CNT : C 15<br>OUT2 : F 58.3 | :A = USTR<br>:SSU = CNT<br>:A = DSTR<br>:SFD = CNT<br>:A = STRT<br>:L KC 100<br>:SEC = CNT<br>:AN = CNT<br>:= = OUT2 | :A I 10.5<br>:CU C 15<br>:A I 10.6<br>:CD C 15<br>:A I 10.7<br>:L KC 100<br>:SE C 15<br>:AN C15<br>:= F 58.3 |
| <b>:JU FB 205</b><br>NAME : <b>EXAMPLE3</b><br>STAR : I 10.4<br>TIME : T 18<br>VALU : IW 20<br>OUT3 : F 100.7                 | :A = STAR<br>:L = VALU<br>:SEC = TIME<br>:A = VALU<br>:= = OUT3  | :A I 10.4<br>:L IW 20<br>:SE T 18<br>:A T 18<br>:= F 100.7   |

• Load and Transfer Operations with Formal Operands

| Operation                                       | Function  |
|---|---|
| L = <input type="text"/>                        | Load a formal operand<br>The value of the operand specified as a formal operand is loaded into the accumulator (parameter type I, Q, T, or C; data type BY, W, D)                 |
| LD = <input type="text"/>                       | Load a formal operand in BCD code<br>The value of the timer or counter register specified as a formal operand is loaded into the accumulator in BCD code. (parameter type T or C) |
| LW = <input type="text"/>                       | Load the bit pattern of a formal operand.<br>The bit pattern of a formal operand is loaded into the accumulator (parameter type D; data type KF, KH, KM, KY, KS, KT, or KC)       |
| LWD <input type="text"/>                        | Load the bit pattern of a formal operand.<br>The bit pattern of a formal operand is loaded into the accumulator (parameter type D; data type KG)                                  |
| T = <input type="text"/><br>↑<br>Formal Operand | Transfer to a formal operand.<br>The contents of the accumulator are transferred to the operand specified as a formal operand. (parameter type I or Q; data type BY, W, or D)     |

For information on programming see section 2.3.2.

Actual operands permitted include those of the corresponding basic operations (except S flags). For the LW operation, permissible constant types include a binary pattern (KM), a hexadecimal pattern (KH), two absolute numbers of one byte each (KY), a character (KS), a fixed-point number (KF), a time value (KT), and a count value (KC). For LWD, permissible constant is a floating-point number.



- Load and Transfer Operations in the RI, RJ, RS, and RT Areas

| Operation | Parameter | Function   |
|-----------|-----------|--|
| L RI      | 0 to 255  | Load a word from the interface data area into ACCU 1 (RI area).                          |
| L RJ      | 0 to 255  | Load a word from the extended interface data area into ACCU 1 (RJ area).                 |
| L RS      | 0 to 255  | Load a word from the system data area into ACCU 1 (RS area).                             |
| L 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).          |
| T 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).             |
| T 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, the RS area has specific sections that you can only read and specific sections you can read or to which you can write (transfer). You can write to RS 60 to RS 63.

See section 8.2.4 for more information.

- **Digital Operations**

| Operation | Function   |
|-----------|--|
| AW        | Combine the contents of ACCU 1-Low and ACCU 2-Low (word operation) through logic AND.          |
| OW        | Combine the contents of ACCU 1-Low and ACCU 2-Low (word operation) through logic OR.           |
| XOW       | Combine the contents of ACCU 1-Low and ACCU 2-Low (word operation) through logic EXCLUSIVE OR. |

Digital operations do not affect ACCU 3 and ACCU 4, but they do affect condition codes CC 1 and CC 0.

Using two load operations, you can load ACCU 1 and ACCU 2 with the corresponding operands of the load operations. Then the contents of both accumulators are combined through digital logic.

**Organizational Functions**

- **Supplementary Jump Operations**

When you use the supplementary jump operations, you indicate the jump destination for unconditional and conditional jumps symbolically. The symbolic address can have a maximum of four characters and must begin with a letter of the alphabet. The symbolic parameter of the jump operation is identical to the symbolic address of the destination operation. When programming, note that the absolute jumping distance should not exceed  $\pm 127$  words. Also note that 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.

**NOTE**

**Jump operation and jump destination must be in the same segment. You can use a symbolic address to label a jump destination. However, you can only use this particular destination label once in any one segment. The JUR jump operation does not have destination labels. For this jump, you specify an absolutely addressed jump destination as an offset parameter.**

| Operation                                   | Function   |
|---|--|
| JU = adr                                    | <p>Jump unconditionally<br/>The jump is executed regardless of conditions.</p>   |
| JC = adr                                    | <p>Jump conditionally<br/>The conditional jump is executed only if the RLO is 1. If the RLO is 0, the operation is not executed and the RLO is set to 1.</p>   |
| JZ = adr                                    | <p>Jump condition: CC 1, CC 0<br/>The jump is executed only if CC 1 is 0 and CC 0 is 0. The RLO is not changed.</p>  |
| JN = adr                                    | <p>Jump condition: CC 1, CC 0<br/>The jump is executed only if CC 1 is not equal to CC 0. The RLO is not changed.</p>  |
| JP = adr                                    | <p>Jump condition: CC 1, CC 0<br/>The jump is executed only if CC 1 is 1 and CC 0 is 0. The RLO is not changed.</p>  |
| JM = adr                                    | <p>Jump condition: CC 1, CC 0<br/>The jump is executed only if CC 1 is 0 and CC 0 is 1. The RLO is not changed.</p>  |
| JO = adr                                    | <p>Jump on Overflow<br/>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.<br/>An overflow occurs when an arithmetic operation exceeds the legal area for a given number representation.</p> |
| JOS = adr                                   | <p>Jump when the OS (stored overflow) condition code is set (OS is 1).</p>   |
| JOS jump distance:<br>- 32768 to<br>+ 32767 | <p>Jump for system software</p>  |

adr = Symbolic Address (Maximum of four characters)

• Shift and Rotate Operations

| Operation | Parameter | Function   |
|-----------|-----------|--|
| SLW       | 0 to 15   | Shift a word (16 bits) to the left. Vacant positions to the right are padded with zeros.                       |
| SRW       | 0 to 15   | Shift a word (16 bits) to the right. Vacant positions 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 (16 bits) 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 (32 bits).  |
| RRD       | 0 to 32   | Rotate to the right (32 bits).   |

Only ACCU 1 is involved in the execution of the shift operations. The parameter portion 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 the SLD, SSD, RLD, and RRD operations, the entire contents of ACCU 1 (32 bits) are involved.

The shift operations are executed regardless of conditions.

The bit that was shifted out last can be scanned via CC 1/CC 0 using jump operations:

| Shifting: Last Bit Shifted | CC 1 | CC 0 | Jump Operation |
|----------------------------|------|------|----------------|
| 0                          | 0    | 0    | JZ =           |
| 1                          | 1    | 0    | JN =<br>JP =   |

Examples:

| STEP 5 Program | Contents of Data Words |
|----------------|------------------------|
| :L DW 52       | KH = 14AF              |
| :SLW 4         |                        |
| T DW 53        | KH = 4AF0              |

| STEP 5 Program | Contents of ACCU 1<br>(in Hexadecimal Code) |
|----------------|---|
| :L ED 0        | 2348 ABCD                                   |
| :SLW 4         | 2348 BCD0                                   |
| :SRW 4         | 2348 0BCD                                   |
| :SLD 4         | 3480 BCD0                                   |
| :SSW 4         | 3480 FB CD                                  |
| :SSD 4         | 0348 0FBC                                   |
| :RLD 4         | 3480 FB CD                                  |
| :RRD 4         | 0348 0FBC                                   |
| :BE            |   |

Applications:

Multiplying by the Third Power  
e.g., New Value = Old Value x 8

```
:L FW 10
:SLW 3
:T FW 10
```

Attention: Do not exceed the positive area limit.

Dividing by the Second Power  
e.g., New Value = Old Value ÷ 4

```
:C DB 5
:L DW 0
:SRW 2
:T DW 0
```

• Conversion Operations

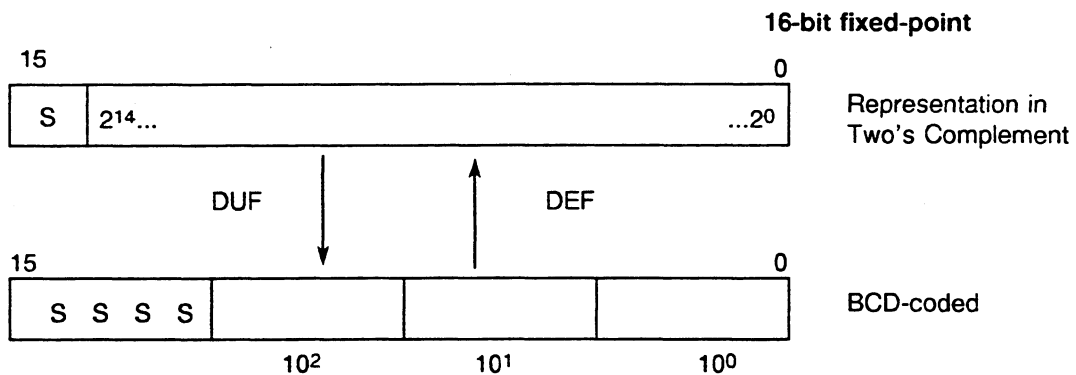
| Operation | Function   |
|-----------|--|
| CFW       | Form the one's complement of ACCU 1-Low (16 bits).                           |
| CSW       | Form the two's complement of ACCU 1-Low (16 bits).                           |
| CSD       | Form the two's complement of ACCU 1-Low (32 bits).                           |
| 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 (32 bits) to a fixed-point number (32 bits). |

**DEF**

The value in ACCU 1-Low (bits 0 to 15) is interpreted as a number in BCD code. After conversion, ACCU 1-Low contains a 16-bit, fixed-point number.

**DUF**

The value in ACCU 1-Low (bits 0 to 15) is interpreted as a 16-bit, fixed-point number. After conversion, ACCU 1-Low contains a number in BCD code.



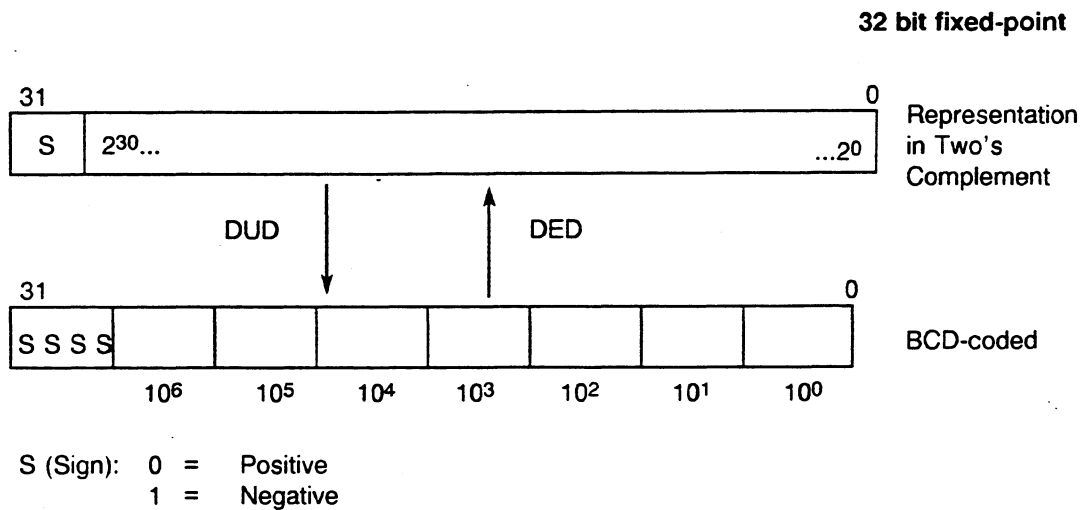
S (Sign): 0 = Positive  
 1 = Negative

**DED**

The value in ACCU 1 (bits 0 to 31) is interpreted as a number in BCD code. After 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 conversion, ACCU 1 contains a number in BCD code.



**FDG**

The value in ACCU 1 (bits 0 to 31) is interpreted as a 32-bit, fixed-point number. After conversion, ACCU 1 contains a floating-point number (exponent and mantissa).

**GFD**

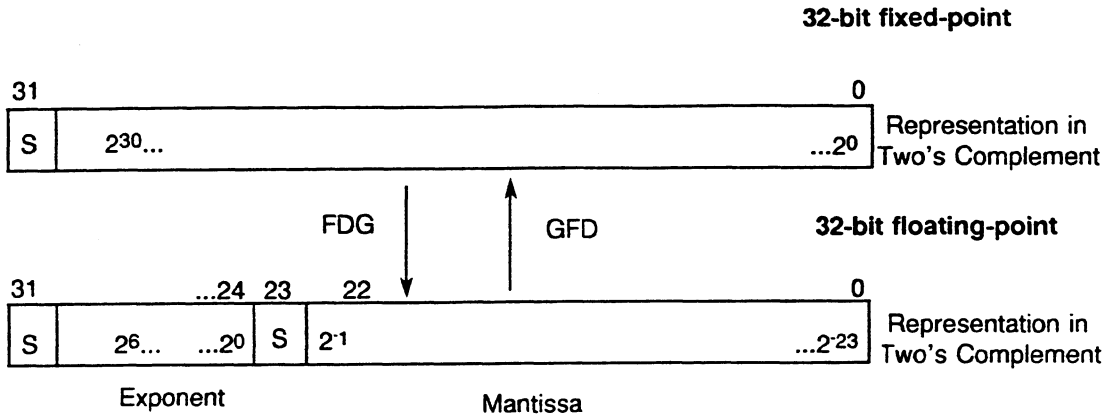
The value in ACCU 1 (bits 0 to 31) is interpreted as a floating-point number. After conversion, ACCU 1 contains a 32-bit, fixed-point number.

The conversion is carried out by multiplying the (dual) mantissa with the value of the (dual) exponent. For this, the mantissa value is shifted to higher-level bit positions via an imaginary decimal point by the value of the exponent (in base '2'). After the multiplication remainders of the original mantissa are left over at the right-hand side of the imaginary decimal point. These bit positions are cut off from the integer result.

The following classes result from this conversion algorithm:

- Floating-point number  $\geq 0$  or  $\leq -1$  result in the next smaller number.
- Floating-point numbers  $< 0$  and  $> -1$  result in the value '0'.

Examples: +5.7 → 5  
 -2.3 → -3  
 -0.6 → 0  
 +0.9 → 0



Examples

a) **CFW** means form 1's complement of ACCU 1-L (bits 0-15).

The contents of data word DW 64 are inverted bit for bit (reversed) and stored in data word DW 78.

| STEP 5 Program | Contents of the Data Words |
|----------------|----------------------------|
| :L DW 64       | KM = 0011111001011011      |
| :CFW           |                            |
| :T DW 78       | KM = 1100000110100100      |

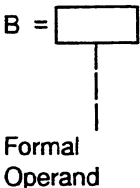
b) **CSW** means form 2's complement of ACCU 1-L (bits 0-15). Result can be evaluated via CC1/CC0 and OV

The contents of data word DW 207 are interpreted as a fixed-point number and stored in data word DW 51 with reversed sign.

| STEP 5 Program | Contents of the Data Words |
|----------------|----------------------------|
| :L DW 207      | KF = + 51                  |
| :CSW           |                            |
| :T DW 51       | KF = - 51                  |



• Other Operations

| Operation   | Parameter | Function  |
|---|-----------|---|
| IA  |           | Disable process interrupt driven processing.  |
| RA  |           | Enable process interrupt driven processing.   |
| IAE   |           | Disable addressing error: bit 4 is reset in the interrupt condition code reset word (UALW).   |
| RAE   |           | Enable addressing error.  |
| BAS   |           | Disable output command: PIQ is no longer affected.  |
| BAF   |           | Enable output command.  |
| D   |           | Decrement ACCU 1-LL by the parameter value.   |
| I   | 1 to 255  | Increment ACCU 1-LL by the parameter value.   |
| ENT   | 1 to 255  | The contents of ACCU 2 are loaded into ACCU 3 and the contents of ACCU 3 are loaded into ACCU 4.  |
| SED   | 0 to 31   | Disable (set) a semaphore (multiprocessing).  |
| SEE   | 0 to 31   | Enable (reset) a semaphore (multiprocessing).   |
| DO  | DW        | 0 to 255<br>Process data word.<br>The operation that follows is executed with the operand whose value is located in the data word of the current DB/DX referenced by the parameter. |
| DO  | FW        | 0 to 254<br>Process flag word.<br>The operation that follows is executed with the operand whose value is located in the flag word referenced by the parameter.                      |
| B =  |           | Process formal operand (parameter type B).<br>You can substitute only C DB, JU PB, JU FB, and JU SB.  |

**Disable/Enable External Process Interrupt Processing (IA, RA)**

If you made a setting in DX 0 to give external process interrupts a higher priority than internal time interrupts, you can use the IA and RA (disable/enable interrupts) operations to suppress external process interrupts when you are using time driven processing (e.g., in a specific OB for time driven program processing). Then external process interrupt driven processing is no longer possible in the program section between the IA and RA operations.

**Decrement/Increment (D/I)**

The low byte of ACCU 1-Low (bits 0 to 7) is decremented or incremented by the number indicated as parameter. The operation is executed regardless of conditions. It is limited to the right byte (without carry).

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 (no carry!)        |

**ENT**

When you use the ENT operation, ACCU 3 and ACCU 4 are also used in arithmetic operations. The ENT operation transfers the contents of ACCU 2 and ACCU 3 to ACCU 3 and ACCU 4. A stack lift takes place as follows:

- <ACCU 4>: = <ACCU 3>
- <ACCU 3>: = <ACCU 2>
- <ACCU 2>: = <ACCU 2>
- <ACCU 1>: = <ACCU 1>

ACCU 1 and ACCU 2 are not changed. The old contents of ACCU 4 no longer exist.

Example

Calculation of the Fraction  $(30 + 3 \times 4)/6 = 7$

|   | ACCU 1 | ACCU 2 | ACCU 3 | ACCU 4 |
|---|--------|--------|--------|--------|
| Assignment of the Accumulators before the Arithmetic Operational Sequence | 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      |
| * F   | 12     | 30     | c      | c      |
| + F   | 42     | c      | c      | c      |
| L KF 6  | 6      | 42     | c      | c      |
| : F   | 7      | c      | c      | c      |

## DO DW and DO FW

You can combine the following operations with the DO DW and DO FW processing operations:

A, AN, O, ON  
 S, R, =  
 L, LD  
 T,  
 FR T, R T, SF T, SD T, SP T, SS T, SE T  
 FR C, R C, S C, CD C, CU C,  
 JU, JC, JZ, JN, JP, JM, JO,  
 SLW, SRW  
 GX DX,  
 D, I,  
 C DB, CX DX, DOU FX, DOC FX, G DB, SED, SEE

The PG does not check if the combinations are permissible. The CPU checks the combinations at the beginning of the processing of the substitution operation.

### Operand Substitution via DO DW/DO FW:

Using the operations "DO DW" and "DO FW" you can access data in a substituted way, e.g. in a program loop. The substituted access consists of the operation DO DW/DO FW in a STEP 5 operation following directly from the operation spectrum mentioned above.

"Substituted" means, that the operand is not statically set for the operation during programming. It is only then defined when you start up your STEP 5 program.

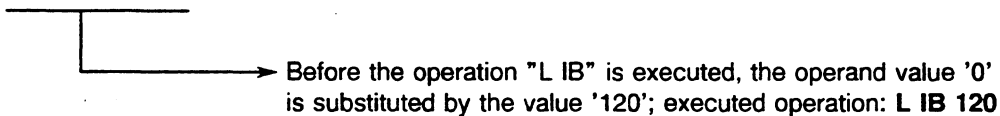
When programming, you must select your operand type from the spectrum permissible for the operation, e.g. PB for the operation "JU PB nn".

Using "DO DW/DO FW" you must load the operand value (nn in the example "JU PB nn") into a data or F flag word (parameter word) prior to a substituted access.

### Examples:

#### 1. Principle of the substitution:

```
:L   KF   + 120
:T   FW   14 Load FW 14 with the value "KF + 120".
:DO  FW   14
:L   IB0
```



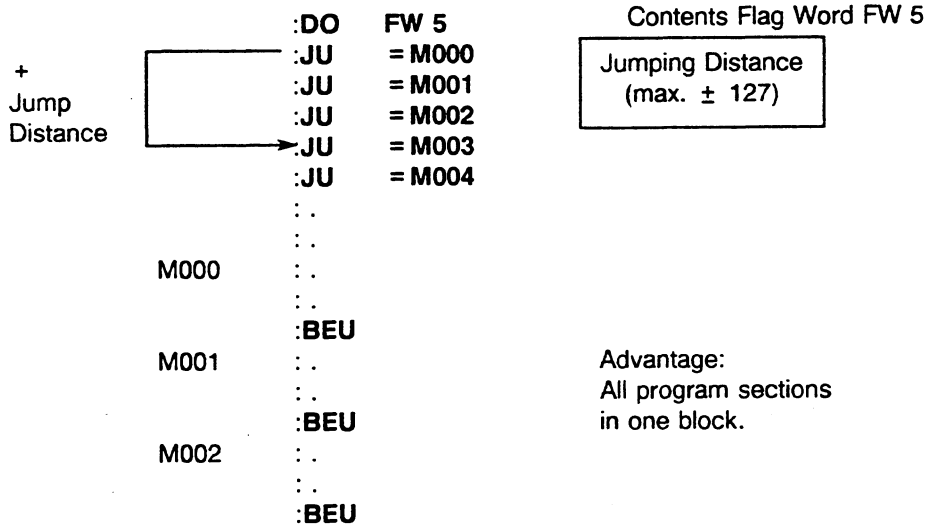
2. Data word as index register

The contents of data words DW 20 to DW 100 are set to signal level 0. The index register for the parameter of the data words is DW 1.

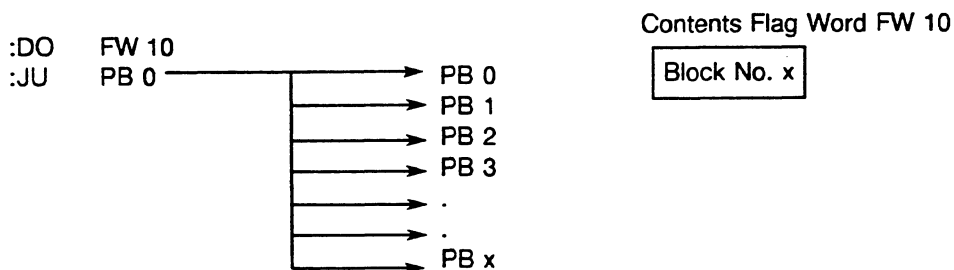
```

M001  :L   KF +20      Supply parameter of the index register.
      :T   DW 1
      :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
    
```

3. Jump distributor for subroutine techniques



4. Jump distributor for block calls

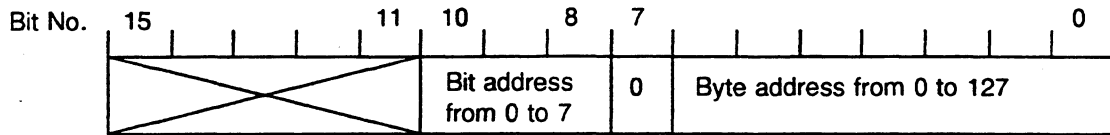


**Operand Substitution with Binary Operations:**

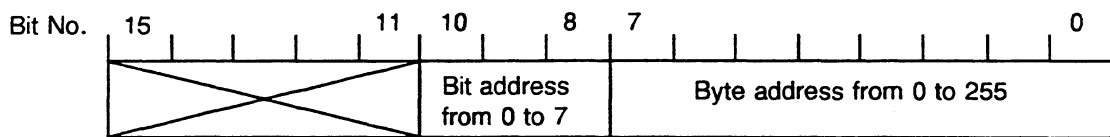
For operand substitutions with binary operations you can use the following operand types: inputs outputs, F flags, S flags, timers and counters.

With these substitutions the design of the F flag or data word (parameter word) depends on the type of operand you use.

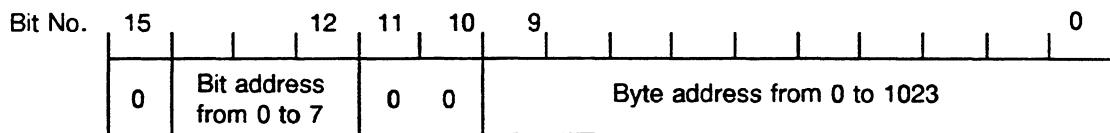
**Parameter Word for Inputs and Outputs:**



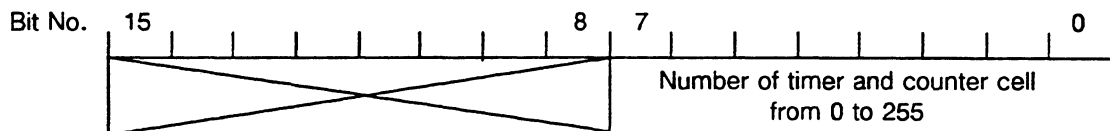
**Parameter Word for F Flags:**



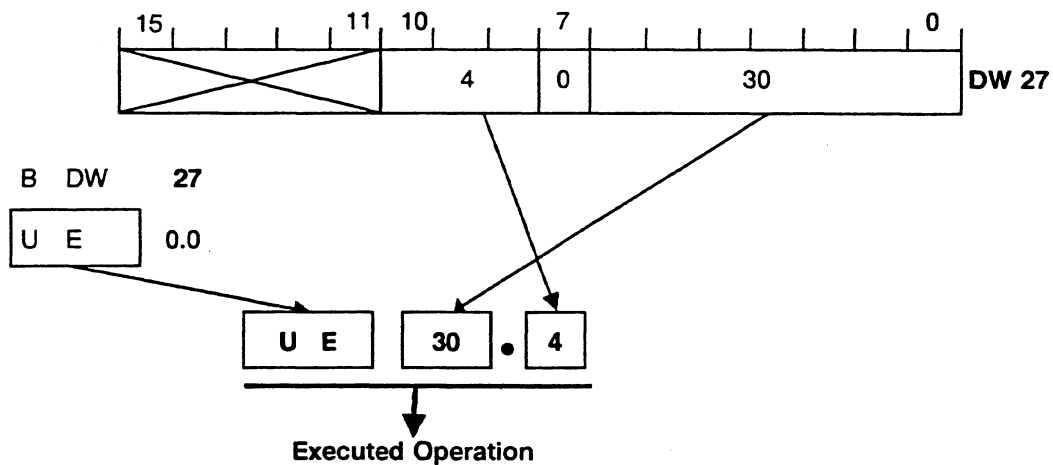
**Parameter Word for S Flags:**



**Parameter Word for Timers and Counters:**



**Principle of the Substitution with a Binary Operation:**

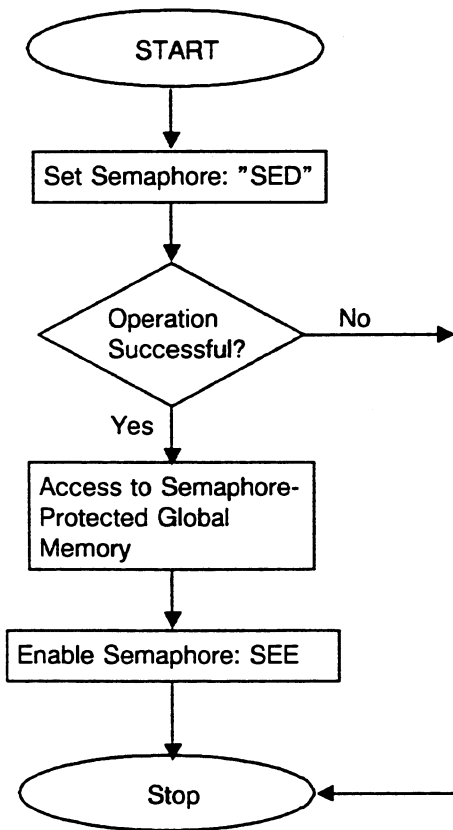


**SED/SEE: Disable/Enable Semaphore (Multiprocessing)**

If two or more CPUs in one programmable controller require read or write access to the same specific global memory area (peripherals, CPs, IPs), you must coordinate CPU access to the common memory area to prevent the following: One CPU might overwrite the data of another CPU. One CPU could read invalid intermediate data statuses of another CPU and misinterpret them.

You can coordinate individual CPU access by using semaphores. A semaphore is a binary variable that can be disabled (set) or enabled (reset) by a CPU. In multiprocessing, all the CPUs share the common memory area on the coordinator. Each CPU must use semaphore operations to access this memory area. The CPU uses the operation SED to disable a semaphore. While the semaphore is disabled, the CPU can access the memory area. When the CPU is finished, it uses the operation SEE to enable the semaphore. The CPU that disables the semaphore is the only one that can enable it again. Once the semaphore is enabled, another CPU has a chance to disable it and access the common memory area. Only one CPU can disable a specific semaphore at a given time. If a CPU cannot disable a semaphore, it cannot access the common memory area.

All CPUs involved in sharing global memory area (multiprocessing) must contain a function block with the program flowchart structure.



Using the operations SED and SEE guarantees that a CPU is protected from interruption by another CPU when transferring related pieces of information into or out of a specific memory area.

**NOTE**

**All CPUs that require access to the same global memory area (addresses greater than F0000H for the CPU 946/947) must synchronize their access by using the operations SED and SEE.**

The CPU that executes the operation SED xx accesses a specific byte in the coordinator (provided that no other CPU has access to that byte already). The operation SED (semaphore disable) sets a semaphore. While this semaphore (0 to 31) is set, it prevents any other CPU from accessing the byte in question. The area is disabled for all other CPUs, and must be programmed as such.

The CPU that executes the operation SEE xx relinquishes its access to a specific byte in the coordinator. The operation SEE (semaphore enable) resets a semaphore. When this semaphore is reset, it enables any other CPU to access the byte in the coordinator. Then one of the other CPUs can write to or read the global memory area. A semaphore can be reset only by the CPU that set it.

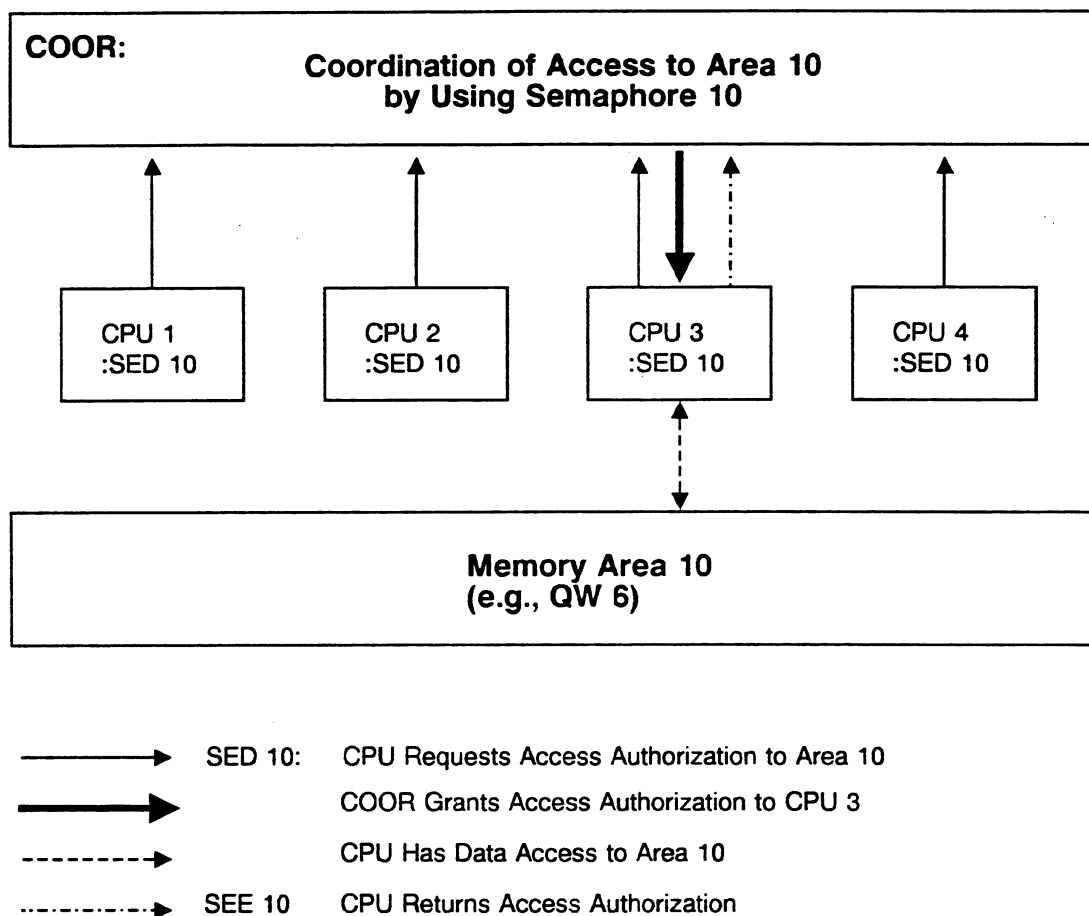


Fig. 3.3 Setting (Enabling) a Semaphore in Multiprocessor Operation

Before a specific semaphore is set or reset, the operations SED and SEE scan the status of the semaphore. The SED operation includes the following procedures:

| CC 1 | CC 0 | Explanation   | Jump Operations for Evaluation |
|------|------|---|--------------------------------|
| 0    | 0    | Semaphore was set by another CPU and cannot be set/reset. | JZ                             |
| 1    | 0    | Semaphore is being set/reset.                             | JN, JP                         |

### NOTE

**Only one CPU at a time can access a semaphore. No other CPU can access the semaphore during the read/write procedures. During this time, access to the S5 bus is not granted to another CPU.**

When using semaphores, please note the following points:

- A semaphore is a global variable (i.e., a semaphore with the number 16 exists **only once** in the entire system even if your controller is using three CPUs).
- **All** CPUs that require coordinated access to a common memory area must use operations SED and SEE.
- All participating CPUs must execute the **same** start-up type. During a cold restart, the system program clears all semaphores. During a manual or automatic warm restart, the system program maintains the semaphores.
- Start-up in multiprocessor operation must be synchronized. Therefore, you **must not** perform any test operation.

### Application Example for Semaphores

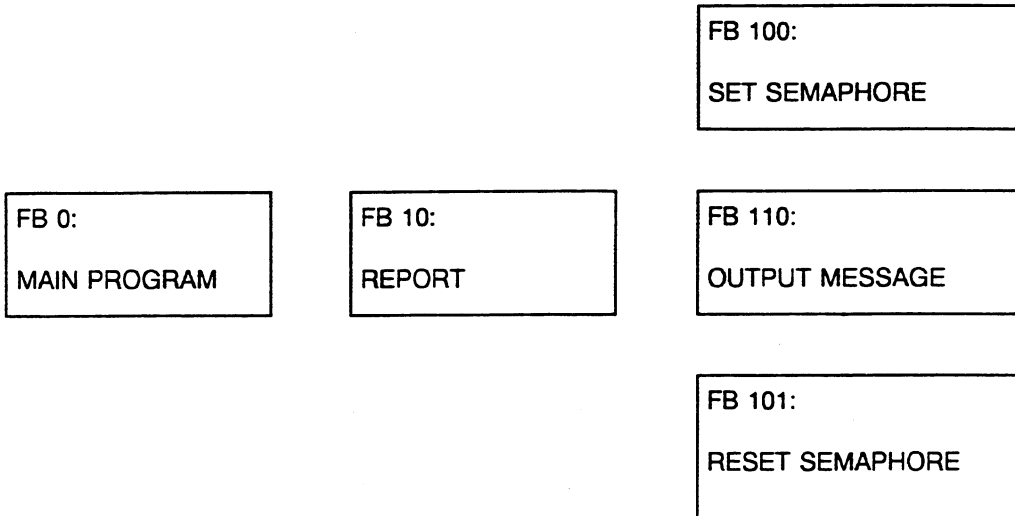
Four CPUs are plugged into an S5-155U. They output dynamic messages to a status reporting device via a common memory area of the O peripherals (e.g., OW 6). A CPU must output each status message for 10 seconds. Only after a 10-second output can a new message from the same CPU or a different CPU overwrite the first message.

This example uses peripheral word OW 6 (extended periphery, no process image update) to be controlled via a semaphore. Only the CPU that was able to reserve this area for itself can write its message to OW 6. The CPU reserves the area by successfully setting the assigned semaphore. The semaphore remains set for 10 seconds at a time (e.g., timer T 10). After the timer runs out, the CPU reenables the semaphore. Reenabling the semaphore enables access to the reserved area for the other CPUs. Therefore, a new message can be written to OW 6.



If one CPU attempts to set a semaphore and the semaphore is already set by a second CPU, the first CPU waits until the next cycle. Then it tries again to set the semaphore and output its message.

The following program can run in all four CPUs, each with a different message. The following blocks are loaded:



Five flags are used as follows:

- F 10.0 = 1: A message was requested or is being processed.
- F 10.1 = 1: The semaphore was set successfully.
- F 10.2 = 1: The timer started.
- F 10.3 = 1: The message was transmitted.
- F 10.4 = 1: The semaphore was reset.

**FB0**

```

NAME      :MAIN
          :A      F 10.0   If no message is active,
          :JC      = M001
          :
          :AN      I 0.0
          :BEC
          :
          :L      KH2222  generate message and
          :T      FW 12
          :AN      F 10.0
          :S      F 10.0   set flag "Message."
          :
M001      :JU      FB 10   Call FB "Report."
NAME      :REPORT
          :
          :BE
  
```

**FB 10**

```
NAME :REPORT
:
:AN    F 10.0      If no semaphore is set,
:JC    FB 100      call FB "Set Semaphore."
NAME :SEMASET
:
:A      F 10.1      If the semaphore is set
:AN     F 10.2      and the timer has not started,
:S      F 10.2
:L      KT010.2    start the timer.
:SE     T 10
:
:A      F 10.2      If the timer has started
:AN     F 10.3      and no message is being transmitted,
:JC     FB 110      call FB "Output Message."
NAME :MSGOUT
:
:A      F 10.2      If the timer has started
:AN     F 10.4      and the semaphore is not reset,
:AN     T 10        and the timer has run out,
:JC     FB 101      call FB "Reset Semaphore."
NAME :SEMARESE
:
:AN     F 10.4      If the semaphore is reset,
:BEC
:
:L      KH0000
:T      FY 10      reset all flags.
:BE
```

**FB 100**

```
NAME : SEMASET
:
:SED    10          Set semaphore number 10.
:JZ     =M001
:AN     F 10.1      If the semaphore is set successfully,
:S      F 10.1      set the flag "Sema-Set."
M001 :BE
```

**FB 110**

```
NAME : MSGOUT
:
:L      FW 12       Transmit a message
:T      OW 6        to the peripherals.
:AN     F 10.3
:S      F 10.3      Set flag "Transmit Message."
:BE
```

**FB 101**

NAME : SEMARESE

:

:SEE 10 Enable semaphore number 10.

:JZ =M001

:AN F 10.4

:S F 10.4 Set flag "Semaphore Reset."

M001 :BE

### 3.4 System Operations

The system operations are similar to the supplementary operations. You can use them only in function blocks. They can only be programmed if the correct setting is made in the PRESETS screen form on the PG.

Chapter 9 describes system operations that work with absolute addresses. Only experienced programmers should use system operations.

- **System Block Call and Jump Operations**

| Operation | Parameter             | Function   |
|-----------|-----------------------|--|
| JUR       | - 32768 to<br>+ 32767 | Jump independently in a function block by a specific number. This operation is always executed regardless of conditions.<br>(Jump distance: -32768 bis +32767) |

- **System Arithmetic Operations**

| Operation | Parameter             | Function  |
|-----------|-----------------------|---|
| ADD BN    | - 128 to + 127        | Add a byte constant (fixed-point) to ACCU 1-Low <sup>1)</sup>   |
| ADD KF    | - 32768 to<br>+ 32767 | Add a fixed-point constant (word) to ACCU 1-Low <sup>1)</sup>   |
| ADD DH    | 0 to<br>FFFF FFFF     | Add a hexadecimal value (double word) to ACCU 1 <sup>1)</sup>   |
| + D       |                       | Add two double-word, fixed-point numbers (32 bits):<br>ACCU 1 + ACCU 2 <sup>2)</sup> .                |
| - D       |                       | Subtract one double-word, fixed-point number from<br>another (32 bits): ACCU 1 - ACCU 2 <sup>2)</sup> |

1) ACCU 2, ACCU 3, and ACCU 4 are not affected.

2) See section on Basic Arithmetic Operations





## Chapter 4

### Modes of Operation

This chapter discusses modes of operation and program processing levels.

#### 4.1 Modes of Operation and Their Program Processing Levels

The CPU 946/947 recognizes the following four modes of operation:

- hard STOP mode
- smooth STOP mode
- RESTART mode
- RUN mode

As described in Chapter 3, the system program calls organization blocks (OB 1 through OB 39) to handle certain conditions. You can specify further reaction of the CPU in these organization blocks (e.g., if a timeout occurs during the update of the process image in the RUN mode, the system program calls OB 24).

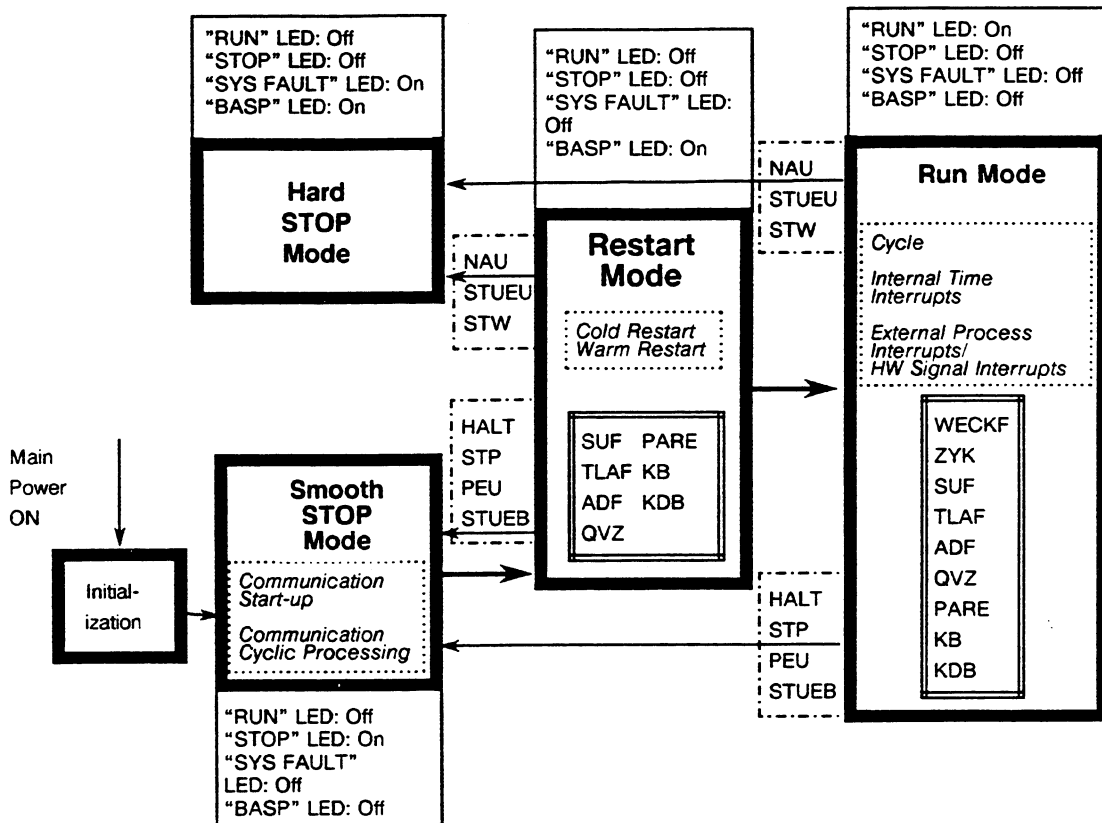
Some conditions can occur only in the RESTART mode and some only in the RUN mode. Others can occur in both the RESTART and RUN modes (see Figure 4-1).

After the system program calls an organization block, the CPU carries out the STEP 5 user program contained in that block. When the CPU does this, it sets up a new register set. A register set consists of a data block start address, a data block length, a STEP address counter, and possibly ACCU 1 to ACCU 4. If an event interrupts normal program processing, the CPU resumes program processing after it processes the OB and all blocks nested in the OB. The CPU resumes program processing at the point where the interruption took place. At this point, the old register contents apply.

A program processing level is assigned to one or more organization blocks. For example, if the system program calls OB 5 in the 150U controller mode, OB 5 activates the external process interrupt program processing level (with "I 0.3").

# Modes of Operation

Figure 4.1 provides an overview of modes of operation and program processing levels.



### Key:

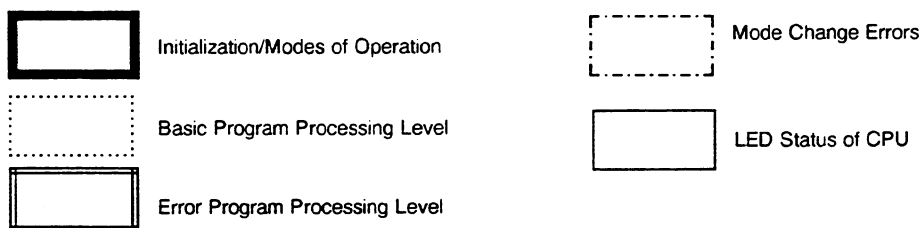


Fig. 4.1 Modes of Operation and Program Processing Levels



The following program processing levels are in the smooth STOP mode (basic levels):

- COMMUNICATION START-UP
- COMMUNICATION CYCLIC PROCESSING

The following program processing levels are in the RESTART mode (basic levels):

- MANUAL COLD RESTART
- AUTOMATIC COLD RESTART
- MANUAL WARM RESTART
- AUTOMATIC WARM RESTART

The following program processing levels are in the RUN mode (basic levels):

- CYCLE (cyclic program processing)
- INTERNAL TIME INTERRUPTS (time driven program processing)
- EXTERNAL PROCESS INTERRUPTS/  
HARDWARE (HW) SIGNAL INTERRUPTS (interrupt driven program processing)

The following program processing levels are in the smooth STOP, RESTART, and RUN modes (error levels):

- collision of time interrupts (WECKF)
- cycle time exceeded error (ZYK)
- substitution error (SUF)
- transfer/load error (TLAF)
- addressing error (ADF)
- timeout (QVZ)
- parity error (PARE)
- called block does not exist (KB)
- opened data block/extended  
data block (DB/DX) does not exist (KDB)

A program processing level has the following **features**:

- Each program processing level has its specific system program.

***Example: System Program of the Cycle Program Processing Level***

In the cycle program processing level, the system program updates the process image input and output tables, triggers the cycle time, and calls the management of the programmer interface (system checkpoint).

- When an interrupt takes place, the system program makes a separate entry in the interrupt stack (ISTACK) for each program processing level. This enables program processing to return to the interrupted level after the interrupt(s).

**Example:**

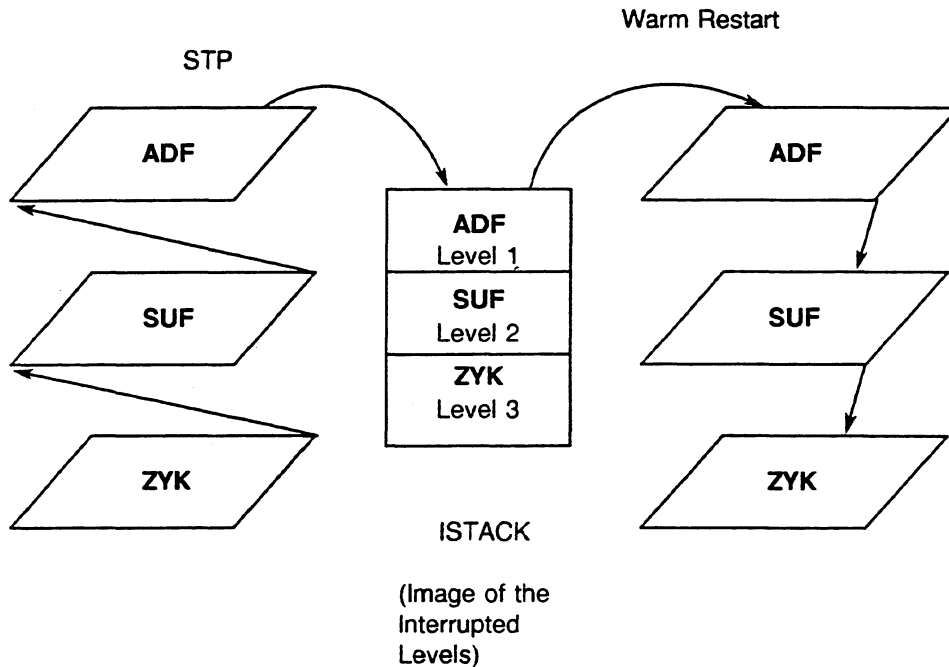


Fig. 4.2 Image of the Interrupted Levels

- Program processing levels can be nested or can interrupt each other.
- A basic level can be nested in a lower priority level at block boundaries (in the 150U controller mode) or at operation boundaries (in the 155U controller mode). An error level (e.g., ADF, QVZ) differs essentially from a basic level in that it is nested immediately at operation boundaries when the event that triggers it occurs.

**Example: Error Level Program Processing in the 150U Controller Mode**

An internal time interrupt occurs during processing of an external process interrupt. Since the default setting gives a time interrupt a higher priority than an external process interrupt, processing of the external process interrupt level is interrupted at the next block boundary and the internal time interrupt program processing level is activated. If an addressing error occurs while the internal time interrupt is being processed, the internal time interrupt is stopped immediately at the next operation boundary to activate the ADF error level.

- The CPU processes the error levels in the order the system program calls them. If additional errors occur during error handling, the system program activates the appropriate program processing levels, lists them, and processes them in sequence.

**NOTE**

Do not nest more than five error organization blocks. If more than five error levels are activated at the same time, an ISTACK overflow occurs and the CPU goes into the STOP mode.

For more information on the individual program processing levels with their related organization blocks, see sections 4.3 and 4.4 and Chapter 5.

Section 4.3 describes the basic levels in RESTART mode.

Section 4.4 describes the basic levels in RUN mode.

Chapter 5 describes the error levels in RESTART and RUN.

## 4.2 STOP Mode

CPU 946/947 has the following two STOP modes: SMOOTH STOP (with communication capability) and HARD STOP

### 4.2.1 SMOOTH STOP

The SMOOTH STOP mode has the following features:

- CPU 946/947 can communicate in the smooth STOP mode. The system program jumps to organization block OB 38 once and then calls organization block OB 39 cyclically. You can call handling blocks and related STEP 5 blocks in these organization blocks.

#### NOTE

**OB 38 is called only when the MAIN POWER is switched ON regardless of the start-up type selected in DX 0 (automatic cold/warm restart).**

- OB 38 is not time monitored. You can abort OB 38 while it is running if you move the RUN/STOP switch on the CPU to the STOP position.

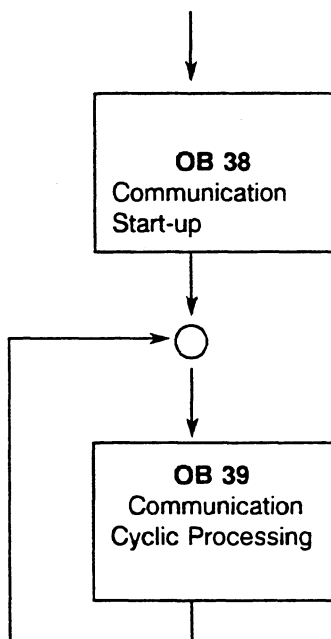


Fig. 4.3 Program Processing after Power-on

- If an interruption in the cyclic program causes the CPU to go into the smooth STOP mode, the system program jumps to OB 39 immediately.
- OB 39 is time monitored. If it runs longer than 2.55 sec., the system program detects a cycle time exceeded error (ZYK) and jumps to OB 26.

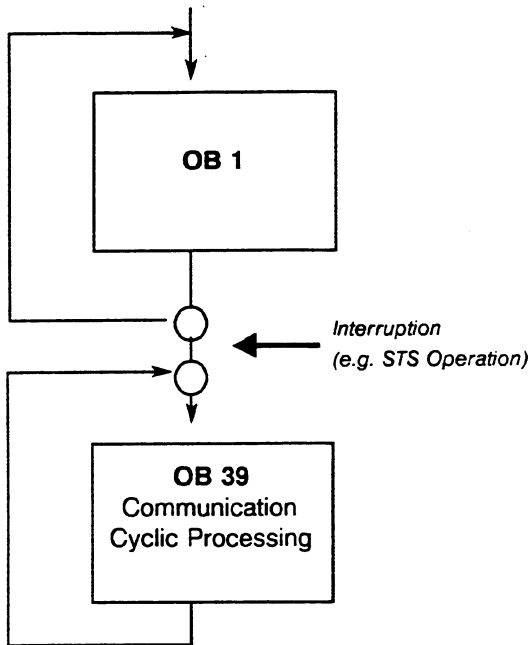


Fig. 4.4 Program Processing following Cycle Interruption

If cyclic program processing is interrupted before a handling block (standard function block) is completely processed, you can jump to the same block or the same type of block (e.g., SEND) in OB 39.

- If an error occurs in OB 39 or in a handling block that has been called there (e.g., ZYK or QVZ), the system program calls the appropriate error organization block, processes it, and then returns to OB 39. If additional errors occur, an ISTACK overflow results if the ISTACK depth exceeds five. The CPU aborts program processing (OB 39 is no longer called) but stays in the smooth STOP mode.
- If cyclic program processing took place, the values of counters, timers, flags, and process images are retained when the CPU goes into the smooth STOP mode.
- The real-time clock continues to run with the 10-ms time base.
- The CPU gives the signal to disable output command (BASP). This disables all digital outputs.

*Exception:* The BASP signal is not output during test operation (see section 10.5).

- Access to the digital peripherals by the CPU is disabled. Attempts to access digital peripherals result in a timeout error (QVZ).
- If cyclic program processing took place before the smooth STOP began, there is an ISTACK in the STOP mode.

| LED       | Condition                     |
|-----------|-------------------------------|
| RUN       | Off                           |
| STOP      | On (constantly or flashing)   |
| SYS FAULT | Off                           |
| BASP      | On (except in test operation) |

Table 4-1 Condition of LEDs in Smooth Stop Mode

The STOP LED indicates the possible causes for any STOP statuses. The following paragraphs describe a continuously lit or flashing STOP LED.

### **STOP LED Stays Lit**

In *single processor operation*,

a constantly lit STOP LED means that the CPU was put into the STOP mode because of one of the following:

- a) The RUN/STOP switch was moved from RUN to STOP.  
Note: The STOP LED does not show a constant light, but flashes slowly, if an error occurred beforehand (eg. addressing error).
- b) The PC STOP programmer function was activated.
- c) A programmable controller fault occurred (e.g., PEU).
- d) An overall reset was performed.

In *multiprocessor operation*,

a constantly lit STOP LED means that the CPU was put into the STOP mode because of one of the following actions originating outside the CPU:

- a) The RUN/STOP switch on the coordinator was moved to STOP.
- b) Another CPU entered the STOP mode because of a fault. (The STOP LED on a CPU that did not cause the fault stays lit.)
- c) The PC STOP function was activated at another CPU.
- d) The END PROGRAM TEST programmer function was activated at another CPU.

### **STOP LED Flashes Slowly**

A slowly-flashing STOP LED (approximately once per second) means that an error occurred.

In *multiprocessor operation*, a slowly-flashing STOP LED indicates the CPU that caused the STOP mode (because of an error).

The CPU was put into the STOP mode because of one of the following:

- a) Stop operation programmed in the user program (STP or STS)
- b) Operator error (e.g., DB 1/DX 0 error, selection of an illegal start-up type)
- c) Block stack (BSTACK) overflow (STUEB) or bracket counter overflow (KZU)
- d) Errors in the user program that have an appropriate error organization block programmed (e.g., call of a block that is not loaded, addressing error, timeout)

The following LEDs light up as an additional indication of the possible cause of error:

- ADF LED
  - QVZ LED
  - ZYK LED
- e) END PROGRAM TEST programmer function at this CPU

### **STOP LED Flashes Quickly**

A quickly flashing STOP LED (approximately twice per second) means that an overall reset is being requested.

### **Requesting an Overall Reset**

Either the system program or you can request an overall reset as follows:

- a) The *system program* requests an overall reset. - Each time you turn on the main power and perform an overall reset, the CPU runs through an initialization routine. If errors are found during this initialization, the CPU goes into the STOP mode and the STOP LED flashes quickly.
- b) *You* request an overall reset

Requesting an Overall Reset

- Move the RUN/STOP switch from RUN to STOP.

Result: The CPU is in the STOP mode.  
The STOP LED stays lit.

- Hold the RESET switch in the OVERALL RESET position. Move the RUN/STOP switch from STOP to RUN and then back to STOP.

Result: The request for an overall reset is completed.  
The STOP LED flashes quickly.

### NOTE

**If you do not want the overall reset that you requested to be carried out, carry out a cold restart. While pressing the RESET button, switch the RUN/STOP switch to RUN.**

### Performing an Overall Reset

Regardless of whether you or the system program requested an overall reset, perform it as indicated:

- Hold the RESET switch in the OVERALL RESET position. Move the RUN/STOP switch from STOP to RUN and then to STOP again.

Result: The overall reset is completed.  
The STOP LED stays lit.

- You can also perform an overall reset by using a programmer. Select the DELETE PC function to perform an overall reset.

Result: The overall reset is completed.  
The STOP LED stays lit.

### NOTE

**All LEDs (except the INIT FAULT LED) are off during the overall reset procedure.**

### NOTE

**After you have performed an overall reset, the only type of start-up possible is a cold restart.**

You can *cancel* the STOP mode in one of the following ways:

- Select a restart type (see section 4.3).
- Perform an overall reset, then perform a cold restart.
- Run the test operation (multiprocessing) (see section 10.5).



#### 4.2.2 HARD STOP

If a system error prevents the system program from operating properly, the CPU goes into the hard STOP mode. This ensures that operation does not continue if the system program has an error.

The following conditions can trigger the hard STOP mode:

- timeout (QVZ) or parity error (PARE) in the system RAM or EPROM
- interrupt stack (ISTACK) overflow (STUEU)
- STEP5 stop operation (STW) for the system program
- serious error in the system program (e.g., firmware fault)

**NOTE**

The CPU is stopped. The hard STOP mode can only be cancelled via power OFF/power ON.

LEDs on the front panel of the CPU in the hard STOP mode.

| LED       | Condition |
|-----------|-----------|
| RUN       | Off       |
| STOP      | Off       |
| SYS FAULT | On        |
| BASP      | On        |

Table 4.2 Condition of LEDs in Hard STOP Mode

### 4.3 RESTART Mode

The RESTART mode is the transition from the STOP mode to the RUN mode. It has the following features:

- The five types of RESTART mode are the following: manual cold restart, automatic cold restart, manual warm restart, automatic warm restart and retentive cold restart.  
You can select or preset these start-up types using one of the following methods:
  - Activate the RUN/STOP and (for cold restart) RESET switches.
  - Use programmer functions.
  - Assign parameters in DX 0.After a cold restart, the system program processes the cyclic user program from the beginning. After a warm restart, user program processing is resumed from the point at which it was interrupted.
- For a manual or an automatic cold restart and for a manual or an automatic warm restart, the system program calls special-organization blocks in which you can program a specific start-up program. These organization blocks are OB 20, OB 21, and OB 22. The length of the STEP 5 start-up program in the OBs is not restricted. It is not time monitored. Other blocks can be called in the start-up OBs.
- Timers are activated before the system program calls start-up organization blocks.
- In each start-up type, the values of counters, timers, flags, and process images are handled differently (see section 4.3.1).
- The disable output command signal (BASP) is active. This disables all digital outputs.  
*Exception:* The BASP signal is inactive during test operation (see section 10.5).
- The interrupts (internal time interrupts, external process interrupts/HW signal interrupts) are disabled.

Note:

For information on the start-up procedure for multiprocessing, see section 10.4.

| LED       | Condition                     |
|-----------|-------------------------------|
| RUN       | Off                           |
| STOP      | Off                           |
| SYS FAULT | Off                           |
| BASP      | On (except in test operation) |

Table 4.3 Start-Up Condition of LEDs in RESTART Mode

During the restart procedure, the INIT FAULT LED on CPU 947 stays lit. If it flashes quickly, there is a system error.

### 4.3.1 Manual and Automatic Cold Restart

#### Manual Cold Restart

Perform a manual cold restart as indicated:

- Hold the RESET switch in the RESET position. Move the RUN/STOP switch from RUN to STOP.
- or
- activate the START programmer function on your PG (select cold restart).

#### Automatic Cold Restart

Perform an automatic cold restart

- The system program performs an automatic cold restart after power goes on (NAU), provided you changed the default in DX 0 to "automatic cold restart after power-on" (see Chapter 7).

The following are prerequisites:           The switches on all CPUs and on the coordinator must remain at RUN.

A cold restart is **always permitted** as long as the system is not requesting an overall reset.

A cold restart is **required** under the following conditions:

- After an overall reset
- After you have inserted EPROM submodules (user program) into the 355 memory module
- After you have loaded your program into the RAM memory (integrated RAM/RAM submodules) in the STOP mode
- After stack overflow
- After cold restart has been aborted (by power failure or the STOP switch)
- After stop with the END PROGRAM TEST programmer function

#### Aborting the Cold Restart

You can abort a running cold restart by moving the RUN/STOP switch to the STOP position. If the main power goes off, this also aborts a running cold restart. You have to repeat an aborted cold restart. You cannot use a warm restart instead.

### 4.3.2 Manual and Automatic Warm Restart

#### Manual Warm Restart

Perform a manual warm restart as indicated:

- Leave the RESET switch in its normal middle position. Move the RUN/STOP switch from STOP to RUN.
- activate the START programmer function on your PG (select warm restart).

A manual warm restart is possible only after one of the following events causes the CPU to go into the STOP mode during cyclic processing:

- The RUN/STOP switch is placed in the STOP position.
- In multiprocessor operation, the coordinator gives the HALT signal.
- The power is shut off (NAU), if you have set the appropriate parameter in DX 0.
- The PC STOP programmer function is activated.

#### NOTE

**If any other event causes the CPU to go into the STOP mode, a warm restart is impossible.**

#### Automatic Warm Restart

To perform an Automatic Warm Restart:

- The system program performs an automatic warm restart after power failure/power OFF (NAU) (when power is turned on again, default setting) as follows:

If the power is turned on again after power down or power failure during cyclic operation, the system program runs through initialization and tries to perform a warm restart automatically.

**The following are prerequisites:**

- The RUN/STOP switch on each CPU and on the coordinator was not moved from the RUN position.
- No other errors occurred during initialization or before NAU.

#### NOTE

**A manual or automatic warm restart is permitted only if the user program was not changed when the CPU was in the STOP mode.**

#### Aborting a Warm Restart

You can abort a running warm restart by moving the RUN/STOP switch to the STOP position or by shutting off the power. With either of these two ways, either a cold or warm restart is possible.

4.3.3 Comparison of Cold and Warm Restart

|                            | Cold Restart   | Warm Restart   |
|----------------------------|--|--|
| Triggered by:              | <p>Moving the RUN/STOP switch from STOP to RUN while the RESET switch is in the RESET position</p> <p>or:</p> <p>PC START programmer function (cold restart)</p> <p>or:</p> <p>Switching on the main power if automatic cold restart is entered in DX 0</p>  | <p>Moving the RUN/STOP switch from STOP to RUN while the RESET switch is in the middle position</p> <p>or:</p> <p>PC START programmer function (warm restart)</p> <p>or:</p> <p>Switching on the main power if the default setting is entered in DX 0 or no DX 0 is programmed.</p>  |
| System program activities: | <ul style="list-style-type: none"> <li>- recognize memory modules</li> <li>- recognize memory submodules (type and size)</li> <li>- construct block address list in DB 0</li> <li>- delete the process image of the inputs</li> <li>- delete the process image of the outputs</li> <li>- delete flags, timers and counters</li> <li>- delete the digital/analog I/O (each 2x128 bytes)</li> <li>- delete interprocessor communication flags (256 bytes)</li> <li>- delete ISTACK/BSTACK</li> <li>- delete semaphores</li> <li>- DB 1 programmed: transfers the digital I/Os and interprocessor communication flag I/Os to the PI lists</li> <li>- DB 1 not programmed: transfers existing modules (only digital I/Os) to the PI lists (interprocessor communication flags are ignored)</li> <li>- Uses settings in DX 0</li> </ul> | <ul style="list-style-type: none"> <li>- check memory modules (as for last cold restart?)<sup>1)</sup></li> <li>- check memory submodules<sup>1)</sup></li> <li>- retain block address list in DB 0</li> <li>- retain the process image of the inputs</li> <li>- retain the process image of the outputs</li> <li>- retain flags, timers and counters</li> <li>- retain interprocessor communication flags</li> <li>- retain ISTACK/BSTACK</li> <li>- retain semaphores</li> <li>- Nothing is transferred from DB 1</li> <li>- Does not evaluate DX 0</li> </ul> |
|                            | <ul style="list-style-type: none"> <li>- Calls user interface <b>OB 20</b> (if programmed)</li> </ul>  | <ul style="list-style-type: none"> <li>- Calls user interface <b>OB 21/OB 22</b> (if programmed)<sup>2)</sup></li> </ul>   |



|                            | Cold Restart   | Retentive Cold Restart  |
|----------------------------|--|---|
| System program activities: | <ul style="list-style-type: none"> <li>- Recognize memory modules</li> <li>- Recognize memory submodules (type and size)</li> <li>- Construct block address list in DB 0</li> <li>- Delete the process image of the inputs</li> <li>- Delete flags, timers and counters</li> <li>- Delete the digital/analog I/O (each 2 x 128 bytes)</li> <li>- Delete interprocessor communication flags</li> <li>- Delete ISTACK/BSTACK</li> <li>- Delete semaphores (all 32)</li> <li>- DB 1 programmed: transfers the digital I/Os and interprocessor communication flag I/Os to the PI lists</li> <li>- DB 1 not programmed: transfers existing modules (only digital I/Os) to the PI lists (interprocessor communication flags are ignored)</li> <li>- Uses settings in DX 0</li> </ul> | <ul style="list-style-type: none"> <li>- Check memory modules (as for last cold restart?) <sup>1)</sup></li> <li>- Check memory submodules <sup>1)</sup></li> <li>- Retain block address list in DB 0</li> <li>- Delete the process image of the inputs</li> <li>- Retain flags, timers and counters</li> <li>- Delete digital I/O (128 byte), retain analog I/O (128 byte)</li> <li>- Retain interprocessor communication flags</li> <li>- Delete ISTACK/BSTACK</li> <li>- Retain semaphore</li> <li>- Nothing is transferred from DB 1</li> <li>- Does not evaluate DX 0</li> </ul> |
|                            | <ul style="list-style-type: none"> <li>- Calls user interface <b>OB 20</b> (if programmed)</li> </ul>  | <ul style="list-style-type: none"> <li>- Calls user interface <b>OB 21/OB 22</b> (if programmed) <sup>2)</sup></li> </ul>   |
|                            | <ul style="list-style-type: none"> <li>- Synchronizes start-up for multiprocessing</li> </ul>  | <ul style="list-style-type: none"> <li>- Synchronizes start-up for multiprocessing</li> </ul>   |
|                            | <ul style="list-style-type: none"> <li>- Transfers to cycle</li> <li>- Disables BASP</li> <li>- Calls OB 1</li> </ul>  | <ul style="list-style-type: none"> <li>- Transfers to cycle</li> <li>- Disables BASP</li> <li>- Calls OB 1</li> </ul>   |

Table 4.5 Comparison of Cold Restart/Retentive Cold Restart

1) Note: If the system program detects data inconsistencies when checking the memory modules and the plug-in submodules during a retentive cold restart, the retentive cold restart is aborted and an error message appears. Afterward, only a cold restart (not retentive!) is possible.

2) With an automatic warm restart, the system program calls the following user organization blocks in the order indicated: OB 38, OB 39, OB 22, and OB 1.

### 4.3.6 Programming the Start-Up Modes

You must store the programs for the start-up modes in the organization blocks OB 20, OB 21 and OB 22. According to the current operation (cold restart etc.) the system program calls one of these blocks when restarting.

#### **Cold Restart: Organization Block OB 20**

When the CPU executes a cold restart, the system program calls up **OB 20 once**. In OB 20, you can store a STEP5 program that can carry out preliminary steps for starting up the cyclic processing before cyclic program processing begins:

- Cold Restart
- set flags
  - start timers
  - prepare data traffic of the CPU with the peripheral modules
  - execute synchronization with CPs (SYNCHRON handling block/blocks, see section 10.2:3)

Terminate OB 20 with the block end statement BE.

After OB 20 is processed, cyclic program processing begins with a call of OB 1.

If OB 20 is not loaded, the CPU begins immediately with cyclic program processing at the end of a cold (after the system activities) restart by calling OB 1.

If you programmed DX 0 to run an automatic cold restart when power is restored, OB 20 is also called.

#### **Manual Warm Restart: Organization Block OB 21**

When the CPU carries out a manual warm restart, the system program calls **OB 21 once**. In OB 21, you can store a STEP5 program that carries out preliminary steps before cyclic program processing resumes.

Terminate OB 21 with the block end statement BE.

After OB 21 is processed, cyclic program processing continues with the next statement after the point at which it was interrupted. The disable output command (BASP) signal remains active while the rest of the cycle is processed. At the end of the remaining cycle, the process image input and output tables are completely cleared and the process peripherals are reset (selectively). The BASP signal is disabled with the beginning of the next complete cycle.

At the end of a manual warm restart, the CPU begins immediately at the point of interruption if OB 21 is not loaded.

If you programmed DX 0 for a retentive cold restart, the system program executes a special **cold restart** after it processes OB 21 (the CPU continues processing with the **first STEP 5 statement in OB 1**). This retentive cold restart maintains the signal status of the flags, the IPC flags, the semaphores and the address list (DB 0).



### Automatic Warm Restart: Organization Block OB 22

When the main power is restored, the CPU tries to resume the interrupted program at the point where it was interrupted.

Initially, the system program calls OB 22 once. In OB 22, you can also store a STEP 5 program that carries out necessary steps before cyclic program processing resumes.

Terminate OB 22 with the block end statement 'BE'.

After OB 22 is processed, cyclic program processing resumes with the next statement at the point where it was interrupted. The disable output command (BASP) signal remains active while the remaining cycle is processed.

At the end of the remaining cycle the PII/PIQ is completely deleted and the process I/O reset.

At the end of an automatic warm restart, the CPU begins immediately after the point of interruption if OB 22 is not loaded.

If you programmed DX 0 for a retentive cold restart, the system program executes a special **cold restart** after it processes OB 22 (the CPU continues processing with the **first STEP 5 statement in OB 1**). This retentive cold restart maintains the signal status of the flags and the address list (DB 0).

### Hot Restart (Extended Automatic Warm Restart) with the CPU 946/947

The hot restart described in IEC standard 65A is also possible with the CPU 946/947. According to IEC 65A, this hot restart is controlled by an external clock. The clock monitors the time between shutting off the CPU and turning it on again. Depending on the length of this time, a restart may be executed.

The system program does not support the hot restart in the CPU 946/947 directly. You must program it.

Available functions include automatic warm restart (OB 22) and an internal real-time clock. This clock is backed up by a battery.

#### Programming a Hot Restart:

The purpose of programming a hot restart is to back up the current time of day regularly (see section 8.2.5) in special memory registers that should be set up for this (e.g., DB/DX). Backup of the current time of day can occur at the end of every cycle or, if great accuracy is required, be time driven by using OB 10.

To calculate the interruption period you must, in OB 22, deduct the time of day in the back-up memory register (= last value before the CPU was switched off) from the current time of day (= first value after the CPU was switched on again).

The time difference is compared to the specified maximum value for the shut-off period. Depending on the result, the warm restart can be allowed or prevented by the stop operation STP.

Depending on the down time, you can also call different function blocks in the OB 22, in which you have programmed reactions to the respective down time.

### 4.3.7 Interruptions in the RESTART Mode

A start-up program can be interrupted by the following:

- power failure
- activating the STOP switch
- program and programmable controller errors

#### Characteristics of an Interrupted Start-Up

The following basic guidelines apply to the start-up of a CPU 946/947:

- If the start-up procedure is interrupted, the next start-up always runs from the very beginning.
- If the cyclic program processing level (or a time driven or interrupt driven program processing level) is interrupted, a warm restart resumes processing from the point of interruption.
- The last start-up type selected always applies  
Example: if the power goes off (NAU) during the cycle and you move the RUN/STOP switch from RUN to STOP, the CPU executes a manual warm restart when power comes back on if you move the RUN/STOP switch to the RUN position again.
- An interrupted cold restart cannot be resumed with a warm restart. The cold restart must be repeated.
- After an interrupted warm restart, either a cold restart or another warm restart is possible.

#### Interrupt Handling by a Warm Restart

The following may cause the CPU to go into the STOP mode: NAU, HALT (multiprocessing), setting the RUN/STOP switch to the STOP position, activating the STOP programmer function. If this occurs while an error block is being processed (e.g., ADF or SUF), and you select a manual warm restart, the CPU finishes processing the error OB before the system program calls OB 21.

A warm restart takes place after handling of the block is completed if the interruption is acknowledged (i.e., the appropriate error organization block does not require the STOP mode). However, only a cold restart is possible if the interruption puts the CPU into the STOP mode.

#### 4.4 RUN Mode

The RUN mode has the following features:

- The user program is processed cyclically.
- All counters and timers started in the program are running. The I/O process images are updated cyclically.
- The disable output command (BASP) signal is inactive. Therefore, all digital outputs are enabled.
- The interprocessor communication flags are being updated cyclically (if interprocessor communication flags have been indicated in DB 1).
- The interrupts (internal time interrupts/external process interrupts/HW signal interrupts) are enabled.

#### Condition of LEDs in RUN Mode

| LED       |     |
|-----------|-----|
| RUN       | On  |
| STOP      | Off |
| SYS FAULT | Off |
| BASP      | Off |

Table 4.6 Condition of LEDs in RUN Mode

#### NOTE

If an automatic or manual warm restart has been executed before the CPU goes from the STOP mode to the RUN mode, the BASP LED stays lit until the rest of the cycle has been processed and the I/O process image has been updated.

#### NOTE

The RUN mode is reached only when the RESTART mode is completed successfully.

The RUN mode has the following three basic program processing levels:

- 1) cyclic
- 2) time driven (by internal time interrupts)
- 3) interrupt driven (by external process interrupts or by HW signal interrupts)

These three levels differ in the following points:

- a) Different events trigger them.
- b) The system program executes different functions in each program processing level.
- c) One or more organization blocks exist as the user interfaces for each program processing level.

### **4.4.1 Cyclic Program Processing (CYCLE)**

Most functions of a programmable controller pertain to **cyclic program processing**.

#### **Triggering**

If the CPU has an error-free cold restart, it begins with cyclic program processing.

#### **System Program Activities**

The system program accomplishes the following:

- At the beginning of the cycle, it sets the cycle time to be monitored. (The cycle time can be preselected in DX 0; default setting is 200 ms).
- It starts cycle-time monitoring.
- It updates the process image input (PII) table.
- It updates the input interprocessor communication flags.
- It calls OB 1 and processes it.
- At the end of the cycle, it updates the process image output (PIQ) table.
- It updates the output interprocessor communication flags.
- It restarts cycle-time monitoring and then begins the cycle again by updating the process image input (PII) table.

#### **User Interface: OB 1**

The system program calls organization block OB 1 as the user interface regularly during cyclic program processing. The system program processes the STEP 5 user program in OB 1 from the beginning to the block end statement BE in OB 1 via all block calls programmed. After the system program activities, the CPU resumes with the first STEP 5 statement in OB 1.

In OB 1, program the calls of program, function, and sequence blocks that are to be processed in the cyclic program.

### Interrupt Locations

Cyclic program processing can be interrupted briefly at block boundaries (150U controller mode) by the following:

- internal time-driven program processing
- external process interrupts (input byte "IB 0")

Cyclic program processing can be interrupted briefly at operation boundaries (155U controller mode) by the following:

- internal time-driven program processing
- HW signal interrupts (INT A/B, E, F, and G)

Cyclic program processing can be interrupted in both controller modes at *operation boundaries* or be completely aborted by the following:

- program error, programmable controller error
- operator action (programmer function or STOP switch setting)

#### 4.4.2 Time-Driven Program Processing (Internal Time Interrupts)

Time driven processing occurs when a time signal (internal time interrupt) from an internal clock causes the CPU to process a specific program, thereby interrupting cyclic program processing. After processing this program, the CPU returns to the point of interruption and resumes processing.

In this way, specific program sections are inserted automatically into cyclic program processing within a specific time base.

#### Triggering

The default setting for the basic clock rate for time driven program processing is 100 ms. You can change it by assigning parameters in DX 0. The permissible value is as follows: basic clock rate =  $yy \times 10 \text{ ms}$  (with  $1 \leq yy \leq FF$ )

It is recommended that you set the basic clock rate to the smallest time unit required for blocks of the cyclic user program.

## Modes of Operation

A clock distributor calls organization blocks in one of the time bases shown below. You can set the time base in DX 0 (see Chapter 7).

|              |          | Time Base 1<br>(Default Setting) |   |     | Time Base 2 |   |     |
|--------------|----------|----------------------------------|---|-----|-------------|---|-----|
| Block called | OB 10 by | basic rate                       | x | 1   | basic rate  | x | 1   |
|              | OB 11    |                                  | x | 2   |             | x | 2   |
|              | OB 12    |                                  | x | 5   |             | x | 4   |
|              | OB 13    |                                  | x | 10  |             | x | 8   |
|              | OB 14    |                                  | x | 20  |             | x | 16  |
|              | OB 15    |                                  | x | 50  |             | x | 32  |
|              | OB 16    |                                  | x | 100 |             | x | 64  |
|              | OB 17    |                                  | x | 200 |             | x | 128 |
|              | OB 18    |                                  | x | 500 |             | x | 256 |

Table 4.7 Call of Organization Blocks in Time-Driven Program Processing

The first OB call after start-up can occur before the time assigned to the OB runs out. For example, if a time constant of 500 s is set for OB 18 (basic clock rate setting in DX 0 is 1 s), initial processing of OB 18 starts approximately 20 s after a cold or warm restart. However, all subsequent calls occur after exactly 500 s.

### User Interface: OB 10 through OB 18

When an internal time interrupt occurs, the organization block (OB 10 through OB 18) whose timer has run out is called as user interface at the next block boundary (150U controller mode) or at the next operation boundary (155U controller mode).

For example, in OB 12, enter the program part that is to be updated in cyclic program processing every 500 ms (using the default setting).


If organization blocks OB 10 through OB 18 are not programmed, cyclic program processing is not interrupted. Time driven program processing does not take place. You can also use DX 0 to disable time driven program processing completely (e.g., for program testing).

### Priorities for Internal Time Interrupts

Priorities for internal time interrupts are as follows:

- a) In individual program processing levels (level priority) - According to the default setting, the internal time interrupt level of program processing has a **higher priority** than external process interrupts/HW signal interrupts.  
However, you can specify the priority sequence of these levels by assigning parameters in DX0.
- b) In internal time interrupt processing (individual priority)

The priority of each internal time interrupt is permanently assigned and corresponds to a specific period number as follows:

|                                  |                          |   |                  |
|----------------------------------|--------------------------|---|------------------|
| Internal time interrupt Period 1 | (OB 10, shortest period) |  | Highest Priority |
| Internal time interrupt Period 2 | (OB 11)                  |   |                  |
| Internal time interrupt Period 3 | (OB 12)                  |   |                  |
| Internal time interrupt Period 4 | (OB 13)                  |   |                  |
| Internal time interrupt Period 5 | (OB 14)                  |   |                  |
| Internal time interrupt Period 6 | (OB 15)                  |   |                  |
| Internal time interrupt Period 7 | (OB 16)                  |   |                  |
| Internal time interrupt Period 8 | (OB 17)                  |   |                  |
| Internal time interrupt Period 9 | (OB 18, longest period)  |   | Lowest Priority  |

The shorter periods are of a higher priority than long ones and can interrupt the longer periods (e.g., OB 12 interrupts OB 17).

Nested processing is possible for the three shortest periods (OB 10, OB 11, and OB 12). For example, if the CPU is processing one OB 10 and a second OB 10 interrupt is triggered, the CPU finishes processing the first OB 10 and then calls the second one immediately. **More than three internal time interrupts** for one of these 3 periods pending simultaneously cause a **collision of these time interrupts**.

### Interrupt Locations

Time driven program processing can be interrupted at block boundaries (150U controller mode) by the following:

- Another internal time interrupt
- One or more external process interrupts (input byte "IB 0") if they are specified in DX 0 as having a higher priority

Time driven program processing can be interrupted at *operation boundaries* (155U controller mode) by the following:

- Another internal time interrupt
- One or more HW signal interrupts (INT A/B, E, F, and G) if they are specified in DX 0 as having a higher priority

Time driven program processing can be interrupted in both controller modes at operation boundaries by the following:

- program error or programmable controller error

### Collision of Time Interrupts

One of the following situations can cause a collision of time interrupts in the CPU 946/947:

- a) There is a queue overflow during the processing of internal time interrupts. For one of the three shortest periods (OB 10, OB 11, and OB 12), more than three internal time interrupts are in the processing queue. One of the other OBs (OB 13 to OB 18) is being called again before its first call has been processed.

Output the interrupt stack (ISTACK) on the programmer, the error code WEFES (collision of software-driven time interrupts) is marked with an "x" in the control bits.

If this error occurs during time driven program processing, the system program calls **OB 33** as the user interface. In OB 33, you can program a specific reaction to this problem. If OB 33 is not programmed, the CPU goes into the "STOP" mode.

If you want program processing to continue after this error occurs, program the block end statement BE in OB 33.

- b) The internal time interrupt clock is masked (ignored) too long (applies to interruptions at block boundaries in the 150U controller mode).

This situation is related to the basic clock rate of the internal time 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 time interrupts occurs.

When you display the interrupt stack (ISTACK) on the programmer, the error code WEFEH (collision of time interrupts caused by the hardware clock) is marked with an "x" in the ISTACK.

The system program calls OB 33 as the user interface. In **OB 33**, you can program a specific reaction to this problem. If OB 33 is not programmed, the CPU continues processing the program.



When the system program calls OB 33, a **code for the collision of time interrupts** is transferred to ACCU 1-Low. It is structured as follows:

- Bit 0 = 1: Queue overflow at internal time interrupt period 1
- Bit 1 = 1: Queue overflow at internal time interrupt period 2
- Bit 2 = 1: Queue overflow at internal time interrupt period 3  
(For OB 10, OB 11, or OB 12, more than three internal time interrupts are in the processing queue.)
- Bit 3 = 1: Queue overflow at internal time interrupt period 4
- Bit 4 = 1: Queue overflow at internal time interrupt period 5
- Bit 5 = 1: Queue overflow at internal time interrupt period 6
- Bit 6 = 1: Queue overflow at internal time interrupt period 7
- Bit 7 = 1: Queue overflow at internal time interrupt period 8
- Bit 8 = 1: Queue overflow at internal time interrupt period 9  
(For OB 13 to OB 18, an OB is being called again before its first call has been processed.)
- Bit 9 = 1: Internal time interrupt clock was masked (ignored) too long

After OB 33 is processed, program processing returns to the interrupted internal time driven OB.

Note:

In the 150U controller mode (interrupt capability at block boundaries), the step address counter (SAC) does not point to the block at whose boundary (BE statement) the collision of time interrupts took place. It points to the block that called the block that caused the error (the return address).

**NOTE**

**For time driven program processing, you can set parameters in data block DX 0 for the following (see Chapter 7):**

- **setting the basic clock rate**
- **setting the clock distributor**
- **setting priorities relative to interrupt driven program processing**
- **disabling internal time interrupt processing**

### 4.4.3 Interrupt Driven Program Processing (External Process Interrupts/HW Signal Interrupts)

The following two types of interrupt driven program processing are possible with CPU 946/947, depending on the controller mode setting: external process interrupt or HW signal interrupts.

#### External Process Interrupts via Input Byte IB0 (Maximum of Eight Interrupts, 150U Controller Mode)

External process interrupt driven processing takes place when a signal level change in input byte "IB0" causes the CPU to interrupt program processing and process a specific program section. After processing this section, the CPU returns to the point of interruption and resumes processing.

#### Triggering

The signal level change of a bit in input byte "IB0" triggers the external process interrupt.

#### User Interfaces: OB 2 through OB 9

The OBs that the system program calls as the user interfaces when an external process interrupt occurs:

|             | <u>Signal Level Change<br/>in Input Byte "IB0" by Bit:</u> |
|-------------|--|
| OB 2 Called | I 0.0  |
| OB 3 Called | I 0.1  |
| OB 4 Called | I 0.2  |
| OB 5 Called | I 0.3  |
| OB 6 Called | I 0.4  |
| OB 7 Called | I 0.5  |
| OB 8 Called | I 0.6  |
| OB 9 Called | I 0.7  |

(e.g., when there is a signal level change of bit "I0.5," the system program calls OB 7).

Enter specific programs in OB 2 through OB 9 that you want to be processed if an external process interrupt occurs.

If the appropriate OB is not programmed, program processing is not interrupted.

#### NOTE

While the system is running, if there is no acknowledgement ("RDY" signal) during an update of input byte "IB0" (e.g., because a digital input module has failed or has a problem), the system program detects a time-out and calls OB 28. If OB 28 is not programmed, the CPU goes into the STOP mode.

### Priorities for External Process Interrupts

Priorities for external process interrupts are as follows:

- a) In individual program processing levels (level priority)

According to the default setting in extended data block DX 0, the external process interrupt program processing level has a lower priority than the internal time interrupts. However, you can change the priority sequence of both levels by assigning parameters in DX 0.

- b) In external process interrupt processing (individual priority)

If several external process interrupts are in the processing queue, the appropriate organization blocks are called according to the following priority sequence:

|                      |       |  |
|----------------------|-------|--|
| OB 2 (high priority) | I 0.0 | ↓<br>Highest Priority<br><br><br><br><br><br><br><br><br>Lowest Priority |
| OB 3                 | I 0.1 |  |
| OB 4                 | I 0.2 |  |
| OB 5                 | I 0.3 |  |
| OB 6                 | I 0.4 |  |
| OB 7                 | I 0.5 |  |
| OB 8                 | I 0.6 |  |
| OB 9 (low priority)  | I 0.7 |  |

No nested processing is possible with external process interrupts. After one external process interrupt OB is completely processed, the system program calls the OB with the next highest priority and processes it. The external process interrupt processing level is exited only after each signal level change in input byte IB0 is considered and the appropriate OBs have been processed.

#### NOTE

**Processing of an external process interrupt cannot be interrupted by a second external process interrupt.**

### Interrupt Locations

External interrupt program processing can be interrupted at block boundaries (150U controller mode) by

- time driven program processing (default setting in DX 0).

It can be interrupted at operation boundaries by

- a program or programmable controller error.

### HW Signal Interrupts via Signal Lines of the S5 Bus (Maximum of Four Interrupts, 155U Controller Mode)

HW signal interrupt driven processing occurs when an S5 bus signal from a digital input module with interrupt capability or from an intelligent input/output module with interrupt capability causes the CPU to interrupt program processing and process a specific program section. After processing this section, the CPU returns to the point of interruption and resumes processing.

#### NOTE

**If interrupt driven processing via S5 bus signal lines is to take place, you must activate the individual HW signal interrupts initially by assigning parameters in DX 0.**

**You must also set the 155U controller mode in DX 0 and enable the HW signal interrupts by closing jumpers on the CPU module.**

#### Triggering

Active status of an interrupt line on the S5 bus triggers the HW signal interrupt. The HW signal interrupt is level triggered (low level). See the instructions for the module triggering the interrupt for how to acknowledge.

#### User Interfaces: OB 2 through OB 5

The OBs that the system program calls as user interfaces when HW signal interrupts occur:

##### Interrupt Triggered by:

|             |   |
|-------------|---|
| OB 2 called | HW signal interrupt X (A or B, slot dependent <sup>1)</sup> ) |
| OB 3 called | HW signal interrupt E   |
| OB 4 called | HW signal interrupt F   |
| OB 5 called | HW signal interrupt G   |

(For more information on HW signal interrupts, refer to the CPU 946/947 Hardware and Installation Guide and the appropriate hardware and installation guides for the peripheral modules used).

The system program calls OB 4, for example, and processes the user program in that block when the HW signal interrupt F occurs.

If the appropriate OB is not programmed, program processing is not interrupted.

<sup>1)</sup> INT A at slot 11, INT B at slot 51.  
(INT C and INT D are available only for other CPU types at slots 91 and 99.)

### Priorities for HW Signal Interrupts

Priorities for HW signal interrupts are as follows:

- a) In individual program processing levels (level priority)

According to the default setting, HW signal interrupt processing is disabled. If you enable it via extended data block DX 0 in the 155U controller mode, the priority of the HW signal interrupt program processing level is lower than the priority of the internal time interrupts. However, you can change the priority sequence of both levels by assigning parameters in DX 0.

- b) In HW signal interrupt processing (individual priority)

If several HW signal interrupts are in the processing queue, the system program calls their appropriate organization blocks according to the priority sequence that is indicated in the parameters assigned in DX 0. You can program priority levels 1 through 5 for the four HW signal interrupts.

After one HW signal interrupt has been completely processed, the system program calls the OB with the next highest priority and processes it. The HW signal interrupt program processing level is exited only after each signal level change of an interrupt line on the S5 bus has been considered and the appropriate OBs have been processed.

#### NOTE

**A HW signal interrupt cannot be interrupted by another HW signal interrupt.**

### Interrupt Locations

HW signal interrupt processing can be interrupted at *operation boundaries* (155U controller mode) by the following:

- other HW signal interrupts
- time driven program processing (default setting in DX 0)

HW signal interrupt processing is interrupted at operation boundaries by the following:

- program error or programmable controller error

### Disabling Interrupt Driven Processing

The system program inserts an interrupt driven program into the cyclic program at a block boundary (150U controller mode) or at a STEP5 operation boundary (155U controller mode).

An interruption of this type can have a negative effect under the following circumstances: if a cyclic program section must be processed within a specific time (e.g., to achieve a specific response time), or if an instruction sequence should not be interrupted (e.g., during reading or writing of related values).

If a section of the user program should not be interrupted by interrupt driven processing, use the following programming procedures:

#### 150U controller mode (external process interrupts via input byte "IB 0")

- Program the user program section in question so it does not contain any block change. Then it cannot be interrupted by an external process interrupt or an internal time interrupt.
- Program the disable external process interrupts (IA) operation. Enable interrupt processing with the enable external process interrupts (RA) operation. No external process interrupt driven program processing is executed between these two operations. The program section that is located between them cannot be interrupted by external process interrupts that occur.

The IA and RA operations are programmable only in function blocks (supplementary operations set).

- You can use special function OB 122 to turn the interrupt disable on or off. OB 122 can disable external process interrupts in a specific part of a program in the same way the IA and RA operations can. OB 122 also disables the internal time interrupts (see section 6.2).

#### 155U controller mode (HW signal interrupts)

- Write the user program section in question in a HW signal interrupt OB and assign this OB the highest priority.
- Use the LIM and SIM (system) operations to read or set the 32-bit HW signal interrupt mask. It is set up like the interrupt condition code reset word (UALW, see section 8.2.4). Each bit represents a specific cause of interrupt. If the appropriate bit is set to 0 (e.g., bit 2<sup>10</sup> for an internal time interrupt), the internal time interrupts are masked (ignored). This means that a registered internal time interrupt remains in the queue, but the running program is not interrupted.
- You can use special function OB 122 to turn interrupt disable on or off. In the 155U controller mode, OB 122 can disable HW signal interrupts and internal time interrupts (see section 6.2).

In DX 0, you can disable interrupt processing completely or individually for HW signal interrupts. However, this is possible only via a cold restart (see Chapter 7).

### **Simultaneous Request from Interrupt and Time Driven Program Processing**

According to the default setting, time driven processing has the **higher priority**. This results in the following behavior of the CPU:

If an internal time interrupt occurs during interrupt driven program processing, the system program interrupts the program at the next interrupt location (block or operation boundary, depending on the controller mode that has been set). The CPU processes the internal time interrupt. After that, interrupt driven program processing is completed.

If an external process interrupt/HW signal interrupt occurs during time driven program processing, time driven program processing is completed first. Then external process interrupt driven or HW signal interrupt driven program processing is started.

If an external process interrupt/HW signal interrupt and an internal time interrupt occur simultaneously, the internal time interrupt is processed first at the next interrupt location. The external process interrupt/HW signal interrupt in the queue is not processed until the CPU finishes processing all internal time interrupts.

If you change the priority sequence of both program processing levels in DX 0 (i.e., giving external process interrupts/HW signal interrupts a higher priority than internal time interrupts), the operation of the CPU changes accordingly after a cold restart.

### **Response Time**

The response time to an external process interrupt/HW signal interrupt corresponds to the processing time of a block (for external process interrupts, 150U controller mode) or a STEP5 operation (for HW signal interrupts, 155U controller mode). However, if internal time interrupts are still in the processing queue when cyclic program processing is interrupted, the interrupt driven program is not processed until all pending internal time interrupts are completely processed. The maximum response time between the occurrence and processing of an external process interrupt/HW signal interrupt increases in this case by the processing time of the internal time interrupts. (You can prevent this by changing the priority in DX 0.)

### **NOTE**

**If you process your program not only cyclically but also time and interrupt driven, you run certain risks. For example, if you insert time driven or interrupt driven processing in your cyclic program, flags used as intermediate flags in your cyclic program could be overwritten when the cycle is interrupted. This happens when the same flag areas are accessed in different program processing levels.**

**For this reason, save the signal levels of the flags in a data block at the beginning of time or interrupt driven program processing and reload them into the flags at the end of the process interruption.**

**Another possibility is to assign flags permanently to separate program processing levels.**

## Chapter 5

# Interrupt and Error Diagnostics

The system program can detect effects of errors in the user program, faulty operation of the CPU, or errors in system program processing.

### 5.1 Frequent Errors in the User Program

This section describes errors that occur most frequently during start-up of the user program. You can avoid these errors easily by doing the following when you write your STEP 5 program:

- When specifying byte addresses for I/Os, make sure that the corresponding modules are plugged into the central controller housing or the expansion unit.
- Make sure that you have provided correct parameters for all operands.
- 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 all data blocks called in the program exist and are long enough.
- Check to see if all blocks called are actually in the memory.
- Be careful when changing function blocks. Check to see that the FBs are assigned the correct operands and that the actual operands are specified.
- Make sure that timers are scanned only once per cycle (e.g., AT 1).



### 5.2 Error Information Analysis

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 find the problem.

#### a) 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.

- STOP            Stays lit
- STOP            Flashes slowly
- STOP            Flashes quickly
- SYS FAULT      Stays lit

The various conditions of the STOP LED indicate specific causes for interruptions and errors (see section 4.2).

Each of the following LEDs on the front panel stays lit because of a **corresponding error**:

- ADF (addressing error)
- QVZ (timeout error)
- ZYK (cycle time exceeded error).

#### b) ISTACK Programmer Function (see section 5.3)

You can get information about the status of the control bits and the contents of the interrupt stack (ISTACK) by using a combination of the PC INFO and ISTACK programmer functions.

When the CPU goes into the STOP mode, the microprocessor program enters all information in the ISTACK, which can be helpful for 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).

**c) BSTACK Programmer Function**

You can display the contents of the block stack (BSTACK) by using a combination of the PC INFO and BSTACK programmer functions after an error occurs and the CPU is in the STOP mode (see section 3.1.1).

The BSTACK contains a listing of all blocks (blocks of the user program and organization blocks of the system program) called in sequence and not completely processed when the CPU went into the STOP mode. The BSTACK can also contain system program OBs that may precede OB 1. Since the BSTACK is filled from the bottom, the block on the uppermost level of the BSTACK display calls the block that caused the error.

In the first line, the information shown below is available.

- 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 next operation to be processed in the calling block
- REL ADDR           Relative address of the next operation to be processed in the calling block  
(REL ADDR = RETURN ADDR {absolute address} minus BLOCK ADDR {absolute address})  
(You can display relative addresses on a programmer by using the key operated switch to activate the INPUT DISABLE mode, switch position →|.)
- DB NO                Number of the last data block opened in the calling block
- DB ADDR            Absolute start address in the program memory of the last data block opened in the calling block (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    | E2B10      | E2C40       | 00130    | 0     |         |
| OB 63    | E0FC0      | E12FA       | 0033A    | 0     |         |
| OB 62    | E0490      | E0CBE       | 0082E    | 0     |         |
| OB 61    | E0010      | E0273       | 00263    | 0     |         |

In the example above, PB 3 called the faulty block at relative address "00008 - 1 = 00007" (i.e., the next operation to be processed in this block is at relative address "00008." During the jump to this faulty block, no data block was open.

### **Summary**

Use all available information when looking for the causes of errors.

This information includes the following:

1. LEDs on the front panel of the CPU - Specific conditions of these LEDs indicate specific causes for errors or interruptions.
2. ISTACK programmer function - Using this function, you can always display the control bits. When the CPU is in the STOP mode, you can also display the ISTACK.
3. BSTACK programmer function - When the CPU is in the STOP mode, you can look in the uppermost level of the BSTACK to determine the block that called the block in which an error occurred.
4. System data register RS 75 - Experienced programmers can read detailed error information from this register (see section 8.2.4).

### 5.3 Control Bits and Interrupt Stack

You can use a combination of the PCINFO and ISTACK programmer functions 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 (first page of the ISTACK function) in any mode. You can display the ISTACK (any pages of ISTACK information that follow the control bits) 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** (i.e., the second page of the ISTACK function and any pages that follow) 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 errors occurred, a multiple level ISTACK is constructed as follows:

DEPTH (level) 01 = last cause of problem  
DEPTH (level) 02 = next to last cause of problem, etc.

When an ISTACK overflow occurs, the CPU goes into the hard STOP mode immediately. Then you must 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 in sections 5.3.1 and 5.3.2.

**CONTROL BITS**

When you display the ISTACK on your programmer, the status of the control bits is indicated on the first page. Example for the first screen page:

| CONTROL BITS         |            |          |           |           |        |        |
|----------------------|------------|----------|-----------|-----------|--------|--------|
| SYSTEM DESCRIPTION : | E0VH<br>X  | GEP<br>X | BATT      | EINP<br>X | MEHRP  | SYNCR  |
|                      | TEST       | 150U     | 155U<br>X |           |        |        |
| STOP CAUSE :         | PGSTP      | HALT     | STP       | STS       | STOPS  | BEARBE |
|                      | UPROG      | USYS     | UANL      | AFEL      | SYSFHL |        |
| START-UP IDs :       | NEUDF<br>X | WIEDF    | URLDF     | NEUZU     | WIEZU  | URLER  |
|                      | AWEG<br>X  | ANEG     | MSEG      |           |        |        |
| ERROR IDs :          | QVZIN      | PARIN    | BSTKF     | BSTEF     | BGRUN  | MODUN  |
|                      | SEPRF      | SRAMF    | UAFEHL    | KDB1      | KDX0   | FDB1   |
|                      | FDX0       | FMODE    | FEDBX     | QVZNIO    | WEFES  | DB0UN  |

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 driven program processing) modes. If several errors occur, all errors are displayed in the control bits.

**EXPLANATION: CONTROL BITS**

- E0VH     Input byte "IB 0" exists for external process interrupts (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  
150U 150U controller mode set  
155U 155U controller mode set

**STOP CAUSE:** (see RS 7)

PGSTP STOP mode set from programmer  
HALT Multiprocessor STOP mode:  
a) selector switch on the coordinator (COOR) is in the STOP position.  
b) another CPU entered the STOP mode in multiprocessing.  
STP a) STOP mode caused by STEP 5 operation STP (at the end of the cycle).  
b) STOP mode caused by STOP command from system program, if the appropriate error OB is not programmed when an error occurs.  
STS STOP mode caused by STEP 5 operation STS (at the end of an operation)  
STOPS STOP mode caused by setting the RUN/STOP switch to the STOP position  
BEARBE STOP mode after the END PROGRAM TEST programmer function  
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 etc.)

**START-UP IDs:** (see RS 8)

NEUDF Cold restart was executed as last start-up type or is active  
WIEDF Warm restart was executed as last start-up type or is active  
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

## *Interrupt and Error Diagnostics*

---

ANEG Automatic cold restart is preset

MSEG Manual start is preset

### **ERROR IDs:**

QVZIN Timeout error in initialization

PARIN Parity error in initialization

BSTKF Wrong block ID

BSTEF Wrong block delimiter

BGRUN Memory module configuration not the same as the configuration during the last cold restart . Therefore, no warm restart possible

MODUN Memory submodule configuration not the same as the configuration during the last cold restart. Therefore, no warm restart possible

SEPRF System EPROM error

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 Wrong controller mode

FEDBX Error in the STEP 5 operations G DB, GX DX

QVZNIO QVZ test faulty

WEFES Collision of software driven time interrupts: queue overflow

DB0UN DB 0 parameter setup not the same as the parameter setup during the last cold restart. Therefore, no warm restart possible

### **ISTACK**

After you display the control bits on your programmer and press <ENTER>, the next page displayed is the ISTACK. 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. Usually a single error is marked under CAUSE OF INTERR. With several errors, the

corresponding number of ISTACK levels are output. The error that occurred directly before the CPU went into the STOP mode is marked in the ISTACK in DEPTH 01 (level 01).  
 Example for the ISTACK display on your programmer:

| ISTACK            |           |            |           |          |           |         |           |
|-------------------|-----------|------------|-----------|----------|-----------|---------|-----------|
| DEPTH 01          |           |            |           |          |           |         |           |
| INS-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      | BS-REG: | 00000     |
| NUMBER:           | 00FD      | SAC(old):  | 000B2     | ICMK:    | 09DF3FBF  | ICRW:   | FFFFFFFF  |
| BRACKETS:         | KE1 000   | KE2 000    | KE3 000   | KE4 000  | KE5 000   | KE6 000 |           |
| ACCU 1:           | 0000 31BA | ACCU 2:    | 0000 0005 | ACCU 3:  | 0004 0005 | ACCU 4: | 0004 0005 |
| CONDITION CODE:   | CC 1      | CC 0       | OVFL      | OVFLS    | OR        | ERAB    |           |
|                   | STATUS    | RLO        |           |          |           |         |           |
|                   | X         | X          |           |          |           |         |           |
| CAUSE OF INTERR.: | KB        | KDB        | TLAF      | SUF      | STUEB     | STUEU   |           |
|                   | NAU       | QVZ        | ADF       | PARE     | ZYK       | STOP    |           |
|                   |           |            |           |          | X         |         |           |
|                   | STS       | WEFEH      | PEU       | HALT     |           |         |           |

**EXPLANATION OF THE ISTACK DISPLAY:**

DEPTH Level of the ISTACK when errors are nested

DEPTH 01 = last cause of stop to occur  
 DEPTH 02 = next to last cause of stop to occur  
 .....

INS-REG Instruction register:  
 Contains machine code (first word) of the instruction processed last in an interrupted program processing level  
 (Decompilation: see machine code listing in the List of Operations)

BLK-STP Block stack (BSTACK) pointer:  
 Contains the 16-bit offset address of the last BSTACK entry (always Exxxx)

PAGE Number of the current dual-port RAM page selected ( dual-port RAM page access  
 NUMBER refers to this dual-port RAM page)

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.



## Interrupt and Error Diagnostics

---

|             |   |
|-------------|---|
| ...-NO.     | Block type and number of the most recently processed block  |
| REL-SAC     | Relative STEP address counter:<br>Contains the <u>relative</u> address (related to the block start address) of the next operation to be processed in the last <u>block</u> processed<br><br>(You can display relative addresses on a programmer by using the key operated switch to activate the "INPUT DISABLE" mode {switch position → } or output the block on a printer.) |
| SAC (old)   | STEP address counter (old):<br>Contains the <u>absolute</u> address of the operation processed last in an interrupted program processing level in the <u>program memory</u> . When an error occurs, the SAC points directly to the operation that caused the error.   |
| DB-ADD      | Absolute start address in the program memory of the data block currently opened (DW 0) (DB-ADD = 0000 if no data block was opened)  |
| DB-NO.      | Number of the data block currently opened   |
| DBL-REG     | Length of the data block currently opened   |
| ICMK        | Interrupt condition code masking word (see section 8.2.4)   |
| BA-ADD      | Absolute address in the program memory for the operation to be processed next in the block where the last block call was made   |
| ...-NO.     | Block type and number of the block that was called last   |
| BS-REG      | Contents of the BR (base address) register before transition to the STOP mode   |
| ICRW        | Interrupt condition code reset word (system data assignment) (see section 8.2.4)  |
| ACCU 1 to 4 | Contents of the calculation registers (accumulators) before the CPU went into the STOP mode   |

**CONDITION CODE:** See Section 3.2

### CAUSE OF INTERR.:

|       |  |
|-------|--|
| KB    | Called block not loaded (error organization block OB 19)   |
| KDB   | Opened data block not loaded (error organization block OB 19)  |
| TLAF  | Load or transfer error during access to data block (DB) or extended data block (DX) (error organization block OB 32) |
| SUF   | Substitution error (error organization block OB 27). Processed STEP 5 operation cannot be substituted                |
| STUEB | Block stack overflow (nesting depth too great; cold restart required)  |

STUEU Interrupt stack overflow (nesting depth too great; cold restart required)

|       |   |
|-------|---|
| NAU   | Power failure in the central controller   |
| QVZ   | Timeout during data exchange with I/O peripherals (error organization block OB 23/OB 24/OB 28/OB 29)  |
| ADF   | Addressing error for inputs and outputs (error organization block OB 25) in the PI  |
| PARE  | Parity error (error organization block OB 30)   |
| ZYK   | Cycle time exceeded (error organization block OB 26)  |
| STOP  | STOP mode caused by setting the RUN/STOP switch to the STOP position  |
| STS   | STOP mode caused by STEP 5 operation STS (at the end of the operation)  |
| WEFEH | Collision of time interrupts caused by the hardware clock (error organization block OB 33): time interrupt clock was masked (ignored) for too long                                |
| PEU   | Power failure in expansion unit: after a statically pending PEU signal is removed (expansion unit is switched on), the system program always calls OB 22 (automatic warm restart) |
| HALT  | Multiprocessor STOP mode:<br>a) selector switch on the coordinator (COOR) is in the STOP position<br>b) another CPU entered the STOP mode in multiprocessing                      |

## 5.4 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
- process a special error handling program

For the following causes of error, OBs are available.

| Cause of Error   | Organization Block Called | Reaction of CPU if OB Is Not Programmed |
|--|---------------------------|---|
| Call of a block that is not loaded   | OB 19                     | none                                    |
| Attempt to open a data block/extended data block that is not loaded                                    | OB 19                     | STOP                                    |
| Timeout in the user program during access to I/O peripherals   | OB 23                     | none                                    |
| Timeout during update of the process image table and during interprocessor communication flag transfer | OB 24                     | none                                    |
| Addressing error   | OB 25                     | STOP <sup>1)</sup>                      |
| Cycle time exceeded  | OB 26                     | STOP                                    |
| Substitution error   | OB 27                     | STOP                                    |
| Timeout by reading input byte "IB 0" (external process interrupts, 150U controller mode)               | OB 28                     | STOP                                    |
| Timeout during access to the distributed I/O peripherals (extended address area)                       | OB 29                     | none                                    |
| Parity error and timeout in the user memory  | OB 30                     | STOP                                    |
| Load and transfer error  | OB 32                     | STOP                                    |
| Collision of time interrupts:  |                           |   |
| a) queue overflow  | OB 33                     | STOP                                    |
| b) time interrupt clock was masked (ignored) too long  | OB 33                     | none                                    |
| Error during STEP 5 operation G DB/G DX  | OB 34                     | STOP                                    |

<sup>1)</sup> The CPU enters the STOP mode only if the addressing error is not disabled by the STEP 5 operation IAE.

Depending on the error, the CPU can react to an error organization block in one of the following ways:

**a) No interruption of cyclic program processing**

If a timeout error occurs and neither OB 23 nor OB 24 is programmed, cyclic program processing is not interrupted. 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) in the appropriate organization block and terminate it with the block end statement BE.

Program for Putting the CPU into the STOP Mode

```
:  
:  
:  
:STP  
:BE
```

**b) CPU enters the STOP mode**

The CPU goes into the STOP mode immediately when a collision of time interrupts or load/transfer error occurs if you did not program the appropriate organization block.

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.

Program for Uninterrupted Operation

```
:  
:  
:  
:BE
```

**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. You can nest a maximum of five error organization blocks. With more than 5 errors, the CPU goes into the STOP mode because of ISTACK overflow.

### 5.5 Causes of Error and Error Organization Blocks

Specific events can interrupt cyclic, time driven, 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. Interruptions can also cause the system program to call the appropriate error organization block. Interruptions during the start-up program are handled like those in the RUN mode.

The following two types of interruptions put the CPU into the STOP mode: interruptions that put the CPU into the STOP mode directly (e.g., NAU and STUEU cause a hard stop, PEU causes a smooth stop) and interruptions you program so that the system program calls specific organization blocks before the CPU goes into the STOP mode (e.g., with QVZI0, the system program calls OB 28; with ADF, it calls OB 25)

If an error occurs, note the entries in the control bits under Error ids and the entries in the ISTACK under CAUSE OF INTERR.

**Possible causes of interruptions and errors and the organization blocks that handle them.**

#### **Substitution Error (SUF), Organization Block OB 27**

If an operation with a formal operand is to be carried out in a function block, the CPU replaces this formal operand with the actual operand in the block when the block is called during user program processing.

The CPU can detect an illegal substitution. It does not execute an operation that can cause a substitution error. The system program interrupts the user program and calls **OB 27**.

You cannot open data blocks DB 0, DB 1, DX 0, and DX 1. The CPU handles the operations C DB0 and C DB1 like substitution errors. A zero is entered in the DBA and DBL registers.

Illegal operation codes also cause a substitution error.

#### **Calling a Block That Is Not Loaded (KB), Organization Block OB 19**

If your program jumps to a block that does not exist, the system program detects an error. This applies to all blocks except data blocks and extended data blocks. It is also true for conditional and unconditional calls. When the system program detects the call of a block that is not loaded, it calls **OB 19**. In OB 19, you can specify how the CPU should proceed.

If OB 19 contains only the block end statement BE, the CPU continues processing the interrupted STEP 5 program. If OB 19 is not programmed, the CPU also continues program processing after a block that is not loaded has been called.

### Calling a Data Block That Is Not Loaded (KDB), Organization Block OB 19

If you call a data block or an extended data block in your program that does not exist in the memory or is marked as invalid, the system program detects an error and calls OB 19. If OB 19 is not programmed, the CPU enters the STOP mode. A zero is entered in the DBA and DBL registers.

Note:

The system program calls OB 19 for either a KB error (programmable logic block not loaded) or a KDB error (data block 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:     0 1 0 1
- for a KDB error:   0 9 0 4

### Load and Transfer Error (TLAF), Organization Block OB 32

When you load data into or transfer data from 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 is maintained.

The system program also detects a load or transfer error if a single bit within a nonexistent data word is to be set/reset or tested. The system program also detects a transfer error if you try to access a data word before you call a data block (using the C DBxxx or CX DXxxx operation).

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. The operation that caused the load or transfer error is not processed. If **OB 32** is not programmed, the CPU enters the STOP mode.

### Addressing Error (ADF), Organization Block OB 25

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 (e.g., the module is not plugged in, it is defective, or it is not defined in 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 result in incorrect or unwanted reactions.

When an addressing error occurs, the system program calls **OB 25**. If OB 25 is not programmed, the CPU enters the STOP mode.

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 the S5-155U List of Operations).

### Timeout Error (QVZ), Organization Blocks OB 23, OB 24, OB 28, and OB 29

A timeout error occurs when an addressable memory area does not respond with the ready signal ("RDY") within a specific time after being addressed. This time is monitored by the hardware (e.g., the time for CPU 946/947 is 150  $\mu$ sec.). 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 organization blocks if they are programmed:

1. Timeout error in the user program during direct access via the S5 bus to a CP, IP, COOR, or to a peripheral module (e.g., with load and transfer operations LPB, LPW, LOB, LOW, TPB, TPW, TOB, or TOW).

The system program jumps to **OB 23**. If **OB 23** is not programmed, processing of the user program continues.

2. Timeout error during update of the process image input/output tables and transfer of interprocessor communication flags:

The system program jumps to **OB 24**. If **OB 24** is not programmed, processing of the user program continues.

A timeout error increases the scan time of the STEP 5 operation in which it occurred if program processing continues (acknowledgement monitoring time plus time for error handling in the system program plus possible processing time for the appropriate error OB).

3. Timeout error at input byte "IB 0" (external process interrupts, only in the 150U mode):  
The system program jumps to **OB 28**. If **OB 28** is not programmed, the CPU enters the STOP mode. Interrupt driven program processing is disabled.
4. Timeout error of the distributed peripherals in the address area FFC00 to FFEFF (interface module IM 302 is defective or is no longer plugged in):  
The system program jumps to **OB 29**. If **OB 29** is not programmed, processing of the user program continues.

When a timeout error occurs, the system program enters the error address in the RS system data area (see Chapter 9) as follows:

RS 68 : QVZ error address high      RS 69 : QVZ error address low

### Parity Error and Timeout Error in the User Memory (PARE), Organization Block OB 30

Possible inconsistencies between a read and write procedure in the user memory are reported to the system program as parity errors. The system program jumps to OB 30. If **OB 30** is not programmed, the CPU enters the STOP mode. The same reaction takes place if a timeout error occurs in the user memory.

The system program enters the error address in the RS system data area as follows:

RS 70: PARE error address high                      RS 71: PARE error address low

### Cycle Time Exceeded Error (ZYG), Organization Block OB 26

The cycle time includes the entire duration of cyclic program processing. One of the following can cause the cycle time set in the CPU to be exceeded: incorrect programming, program loop in a function block, failure of the clock pulse generator.

When a cycle time exceeded error (ZYG) occurs, the system program interrupts the user program and calls **OB 26**. This retriggers cycle-time monitoring.

If OB 26 is not programmed, the CPU enters the STOP mode.

You can set the cycle monitoring time individually by programming OB 31 (see Chapter 7) or by making an entry in extended data block DX 0. The default setting is 200 ms.

### Collision of Time Interrupts Error (WEFES, WEFEH), Organization Block OB 33

Time driven program processing is handled by organization blocks OB 10 to OB 18.

Internal time interrupts have the following features (see also section 4.4.2):

- Shorter periods have a higher priority than longer periods, and they can interrupt longer periods.
- Nested processing is possible for the three shortest periods (OB 10, OB 11, and OB 12).

However, if **more than three internal time interrupts** are in the processing queue simultaneously for one of these three periods, the system program detects a **collision of time interrupts**.



## Interrupt and Error Diagnostics

---

There are two types of time collision errors in the CPU 946/947:

- a) There is a queue overflow during the processing of internal time interrupts. For one of the three shortest periods (OB 10, OB 11, and OB 12), more than three internal time interrupts are in the processing queue or one of the other OBs (OB 13 to OB 18) is being called again before its first call has been processed.

When you display the interrupt stack (ISTACK) on the programmer, the error code WEFES is marked with an "x" in the control bits.

If this error occurs during time driven program processing, the system program calls **OB 33** as the user interface. In OB 33, you can program a specific reaction to this problem. If OB 33 is not programmed, the CPU goes into the STOP mode.

If you want program processing to continue after this error occurs, program the block end statement BE in OB 33.

- b) The internal time interrupt clock is masked (ignored) too long (applies to interruptions at block boundaries in the 150U controller mode).

This situation is related to the basic clock rate of the internal time 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 time interrupts occurs.

When you display the interrupt stack (ISTACK) on the programmer, the error code WEFEH is marked with an "x" in the ISTACK.

The system program calls **OB 33** as the user interface. In OB 33, you can program a specific reaction to this problem. If OB 33 is not programmed, the CPU continues processing the program.

When the system program calls OB 33, a **code for the collision of time interrupts** is transferred to ACCU 1-Low. It is structured as follows:

- Bit 0 = 1: Queue overflow at internal time interrupt period 1  
Bit 1 = 1: Queue overflow at internal time interrupt period 2  
Bit 2 = 1: Queue overflow at internal time interrupt period 3  
    → (For OB 10, OB 11, or OB 12, more than three internal time interrupts are in the processing queue.)  
Bit 3 = 1: Queue overflow at internal time interrupt period 4  
Bit 4 = 1: Queue overflow at internal time interrupt period 5  
Bit 5 = 1: Queue overflow at internal time interrupt period 6  
Bit 6 = 1: Queue overflow at internal time interrupt period 7  
Bit 7 = 1: Queue overflow at internal time interrupt period 8  
Bit 8 = 1: Queue overflow at internal time interrupt period 9  
    → (For OB 13 to OB 18, an OB is being called again before its first call has been processed.)  
Bit 9 = 1: Internal time interrupt clock was masked (ignored) for too long

After OB 33 is processed, program processing returns to the interrupted internal time driven OB.

Note:

In the 150U controller mode (interrupt capability at block boundaries), the step address counter (SAC) does not point to the block at whose boundary (BE statement) the collision of time interrupts took place. It points to the block that called the block that caused the error (return address).

**NOTE**

**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.**

## Chapter 6

# Integrated Special Function Organization Blocks

This chapter describes integrated special functions. They are contained in organization blocks (OBs) as a permanent component of the system program in CPU 946/947. You can call these functions, but you cannot read or change them.

You can activate these special functions as needed by using a conditional (JC OBx) or an unconditional (JU OBx) block call operation.

### Integrated Special Function Overview - CPU 946/947

|                       |   |
|-----------------------|---|
| OB 121                | Set/read time of day  |
| OB 122                | Turn interrupt disable on/off                                       |
| OB 124                | Delete STEP 5 blocks  |
| OB 125                | Generate STEP 5 blocks  |
| OB 126                | Transfer process image tables                                       |
| OB 200,<br>202 to 205 | Functions for multiprocessor communication                          |
| OB 223                | Compare CPU restart types in multiprocessor operation               |
| OB 254, 255           | Transfer/duplicate data blocks (DBs) and extended data blocks (DXs) |

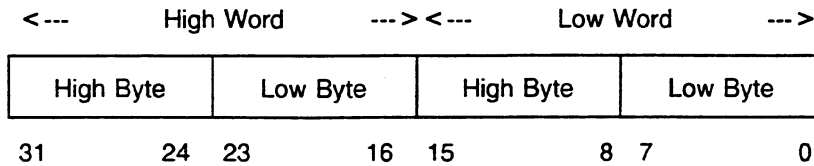
## Integrated Special Function Organization Blocks

---

### Note:

This chapter explains how to assign parameters to individual special function organization blocks (SF OBs).

|            |                              |         |
|------------|------------------------------|---------|
| 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  |



### Note:

In this chapter, the term **parameters** refers to all data that the CPU needs to carry out the special functions correctly. Before you call these special functions in your STEP 5 program, you must load this data into the accumulators or into the memory registers as indicated.

**Errors during Special Function Processing**

Errors that are specifically related to the special functions of CPU 946/947 can occur while these functions are being processed. The special functions affect the result of logic operation (RLO) or condition codes (CC 1/CC 0) so that you can handle such errors.

If an error occurs during processing of these special functions, in most cases the **RLO** is set to 1.

You can use a conditional jump operation (JC) to evaluate the RLO as a signal for error or correct processing. In the event of an error, you can provide a special error handling program.

Processing of some special functions affects condition codes **CC 1** and **CC 0**. You can also scan the condition codes for indications of correct processing.

In addition, after a special function has been called and processed, an ID is indicated in **ACCU 1-Low**. You can also use this ID to determine if the function was processed properly or if an error occurred. If an error occurred, the exact cause of error is given.

| OB Number | Error Evaluation Possible with: |           |            |
|-----------|---------------------------------|-----------|------------|
|           | RLO                             | CC 1/CC 0 | ACCU 1-Low |
| OB 121    | -                               | -         | x          |
| OB 122    | -                               | -         | x          |
| OB 124    | x                               | x         | x          |
| OB 125    | x                               | x         | x          |
| OB 126    | x                               |           | x          |
| OB 223    | x                               | -         | x          |
| OB 254    | x                               | x         | x          |
| OB 255    | x                               | x         | x          |



Cont.:

Hours, tens position: 0 to 1 or 0 to 2  
 Bit 15 = 1: 24-hour format  
 Bit 15 = 0: 12-hour format  
 Bit 14 = 0: AM  
 Bit 14 = 1: PM

Day of week: 0 to 6 (Monday to Sunday)  
 Date, units position: 0 to 9  
 Date, tens position: 0 to 3

Month, units position: 0 to 9  
 Month, tens position: 0 to 1  
 Year, units position: 0 to 9  
 Year, tens position: 0 to 9

**NOTE**

**You must load the start address of the parameter block into the BR register before calling OB 121.**

The following errors are possible:

- illegal function number in ACCU 1-Low
- illegal values in the parameter block for set time of day

If the system program detects an illegal function number in ACCU 1-Low, program processing continues and the system program stores an error ID number in ACCU 1-Low. If the system program detects an illegal value in the parameter block, it also stores an error ID number in ACCU 1-Low. This error ID number explains the error more exactly.

| Error ID Number<br>in ACCU 1-Low | Illegal Parameter<br>for the: |
|----------------------------------|-------------------------------|
| F001H                            | Function number in ACCU 1-Low |
| F101H                            | Year                          |
| F102H                            | Month                         |
| F103H                            | Date                          |
| F104H                            | Day of week                   |
| F105H                            | Hours                         |
| F106H                            | Minutes                       |
| F107H                            | Seconds                       |

If OB 121 is processed correctly, the system program stores the value 0 in ACCU 1-Low.

## Integrated Special Function Organization Blocks

---

### Programming Examples for OB 121

- a) FB 13 is programmed for the function for reading the time of day. Data block DB 10 (data words DW 0 to DW 3) is provided to store the date and time of day.

#### STEP 5 Program:

```
FB 13
NAME  : READTIME
      :MBR      EEC00      Load the start address of the DB list into the BR register.
      :LRW      + 10      Load the start address of DB 10 in the memory (paragraph
      :          :        address) into ACCU 1.
      :L        KB 0
      :!= F
      :JC       = FAIL    Jump to error handling program if the DB start address
      :          :        equals 0.
      :TAK
      :SLD      4         Shift the contents of ACCU 1 to the left by four positions to
      :          :        get the absolute address of DB 10 (DW 0).
      :MAB
      :L        KB 2      Load the contents of ACCU 1 into the BR register.
      :JU       OB 121    Load function number 2 into ACCU 1-Low.
      :BEU
      :          :        Read time of day.
      :          :        Error handling program.
      :          :
      :          :
      :BE
```

OB 121 transfers the current date and time of day in DB 10.

- 0: KH = 2994;      Seconds tens position, seconds units position, 0.1 sec., 0:01 sec.
- 1: KH = 9557;      Hours tens position, hours units position, minutes tens position, minutes units position
- 2: KH = 2810;      Date tens position, date units position, day of week, 0
- 3: KH = 8902;      Year tens position, year units position, month tens position, month units position
- 4:

The current date and time of day is Tuesday, February 28, 1989, 15:57 o'clock, 29 sec., and 940 ms.

- b) FB 14 is programmed for the function for setting the time of day. The new values are transferred in data words DW 0 to DW 3 of data block DB 11.

#### STEP 5 Program:

```
FB 14
NAME  : SETTIME
      :C        DB 11      Open DB 11.
      :L        KH 1500    Load the hexadecimal value for 15 seconds.
      :T        DW 0
      :L        KH 9555    Load the hexadecimal value for 24-hour format, 15:55.
      :T        DW 1
      :L        KH 2810    Load the hexadecimal value for the 28th, Tuesday.
      :T        DW 2
      :L        KH 8902    Load the hexadecimal value for 1989, February.
      :T        DW 3
```



|       |        |   |
|-------|--------|---|
| :MBR  | EEC00  | Load the start address of the DB list into the BR register.   |
| :LRW  | + 11   | Load the start address of DB 11 in the memory (paragraph address) into ACCU 1.                          |
| :     |        |   |
| :SLD  | 4      | Shift the contents of ACCU 1 to the left by four positions to get the absolute address of DB 11 (DW 0). |
| :     |        |   |
| :MAB  |        | Load the contents of ACCU 1 into the BR register.   |
| :L    | KB 1   | Load function number 1 into ACCU 1-Low.   |
| :JU   | OB 121 | Set time of day.  |
| :L    | KB 0   |   |
| :> <F |        | If the contents of ACCU 1 are not equal to 0,   |
| :JC   | = ERRO | jump to error handling program.   |
| :BEU  |        |   |
| :     |        |   |
| ERRO  | :      | Error handling program.   |
| :     |        |   |
| :     |        |   |
| :     |        |   |
| :BE   |        |   |

DB11

|    |            |
|----|------------|
| 0: | KH = 1500; |
| 1: | KH = 9555; |
| 2: | KH = 2810; |
| 3: | KH = 8902; |
| 4: |            |

OB 121 transfers desired date and time of day from DB 11 to the real-time clock hardware block.

### 6.2 Turn Interrupt Disable On/Off (OB 122)

A STEP 5 program can be interrupted by a program of higher priority. The controller mode in which CPU 946/947 is running determines the location of such an interruption, as follows: 150U controller mode - interruption possible at block boundaries, 155U controller mode - interruption possible at operation boundaries. The higher-priority program processing levels include the following: external process interrupts, HW signal interrupts, internal time interrupts. The scan time of a program that is interrupted increases by the scan time of each program that interrupts it.

You can use OB 122 to prevent insertion of interrupts (external process interrupts, HW signal interrupts, internal time interrupts) at one or more consecutive block or operation boundaries.

OB 122 affects the *reception* of interrupts as follows:

Turn on disable interrupts - Effective immediately, no more interrupts are registered.

Turn off disable interrupts - Effective immediately, all interrupts that occur are registered, inserted at the next block or operation boundary (depending on the controller mode set), and processed.

Assign one parameter as follows:

1. ACCU 1-Low: Function ID  
The possible values for the function ID are as follows:  
"1" = disable all interrupts, or  
"2" = enable all interrupts

Error:

- The only possible error is an illegal function ID in ACCU 1-Low.

In the event of an error, the error ID **F001H** is indicated in ACCU 1.

Example:

```

: L KB 1           ;Load the function ID into ACCU 1.
: JU OB 122       ; Disable all interrupts.
:                 ;
: ..              ;
: ..              ;
: ..              ;
: ..              ;
: L KB 2           ;Load the function ID into ACCU 1.
: JU OB 122       ;Enable all interrupts.

```

} Critical part of program

Calling OB 122 affects the RLO. The BR register is not changed. Instead of using OB 122, you can use the operations IA and RA to disable and enable external process interrupts.

### 6.3 Delete STEP 5 Blocks (OB 124)

You can use OB 124 to delete any STEP 5 block (programmable logic or data block) in the user memory. The deleted block is removed from the address list in DB 0. The memory areas vacated by the deletion are reused when new blocks are loaded.

Assign the following two parameters:

1. ACCU 1-LH - *type* of block to be deleted
2. ACCU 1-LL - *number* of block to be deleted

The following block types and numbers are permissible:

| Block Type | Block Number |
|------------|--------------|
| 1 (PB)     | 0 to 255     |
| 2 (SB)     | 0 to 255     |
| 3 (FB)     | 0 to 255     |
| 4 (FX)     | 0 to 255     |
| 5 (DB)     | 1 to 255     |
| 6 (DX)     | 0 to 255     |
| 7 (OB)     | 0 to 39      |

The following operational sequence deletes DX 100 in the user memory:

```
:L KY 6,100
:JU OB 124
```

If OB 124 is processed correctly, the RLO is set to 0. Condition codes CC 1 and CC 0 are also cleared.

In the following cases, an error aborts processing of OB 124:

- The block type or number in ACCU 1-Low is illegal.
- The specified STEP 5 block is in the EPROM and therefore cannot be deleted.
- The block to be deleted does not exist in the user memory.
- The online programmer function COMPRESS MEMORY is running. Simultaneous processing of OB 124 is illegal.

In case of an error, the processing of the SFB is aborted. Program processing is continued with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1-Low (see next page).

## Integrated Special Function Organization Blocks

In the following cases, processing of OB 124 is aborted and a warning given:

- An online programmer function that is in a critical area is running simultaneously (e.g., STATUS).
- OB 124, OB 125, OB 254, or OB 255 was called within the last 10 ms. (Only one of these four special functions can be called within 10 ms. This prevents the interface to the programmer from being cut off completely by multiple calls of these special function OBs.)

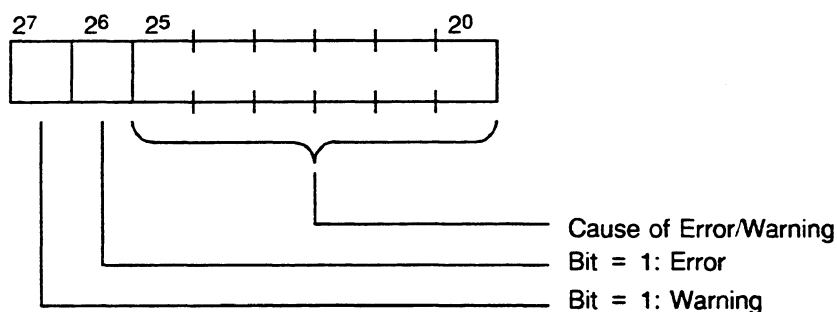
In the event of a warning, program processing is continued with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1 (see below). If a warning aborts OB 124, you can call this special function again to achieve correct processing. It may be necessary to repeat your call several times.

### Affecting the RLO and Condition Codes CC 1 and CC 0

After calling OB 124, you can use the RLO and condition codes CC 1 and CC 0 to determine whether the special function ran correctly or was aborted with an error or a warning. You can evaluate the result with conditional jump operations.

| RLO | CC 1 | CC 0 | Explanation  | Jump Operations for Evaluation |
|-----|------|------|--|--------------------------------|
| 0   | 0    | 0    | The special function was processed correctly.                  | JC                             |
| 1   | 1    | 0    | Processing of the special function was aborted with a warning. | JC, JP                         |
| 1   | 0    | 1    | Processing of the special function was aborted with an error.  | JC, JM                         |

### Location of error IDs in ACCU 1-LL



Causes of errors or warnings during the processing of OB 124:

| Status/Error ID<br>Numbers in ACCU 1-LL | Explanation   |
|---|---|
| 01H                                     | The function was processed correctly.                         |
| 45H                                     | The block type is illegal.                                    |
| 46H                                     | The block is in the EPROM.                                    |
| 47H                                     | The block does not exist.                                     |
| 4DH                                     | The online programmer function COMPRESS<br>MEMORY is running. |
| 8DH                                     | There is a conflict with another online function.             |
| 8EH                                     | The 10 ms waiting time has not elapsed.                       |

With reference to the processing of OB 124, please note the following:

- During the actual deletion procedure, the following user interrupts are disabled: internal time interrupts, external process interrupts (150U controller mode), HW signal interrupts (155U controller mode)
- Calling OB 124 changes the contents of ACCU 1 to ACCU 4. The BR register does not change.

## 6.4 Generate STEP 5 Blocks (OB 125)

You can use OB 125 to generate any STEP 5 block (programmable logic and data block) in the user memory. The specified block is set up in the RAM submodule with block header and block body. Then the block is entered in the address list in DB 0.

The block body contains random data. For this reason, you must write to a newly generated block first. Only then can usable data be read out.

Assign the following three parameters:

1. ACCU 1-LH - *type* of block to be generated
2. ACCU 1-LL - *number* of block to be generated

| Block Type | Block Number |
|------------|--------------|
| 1 (PB)     | 0 to 255     |
| 2 (SB)     | 0 to 255     |
| 3 (FB)     | 0 to 255     |
| 4 (FX)     | 0 to 255     |
| 5 (DB)     | 1 to 255     |
| 6 (DX)     | 0 to 255     |
| 7 (OB)     | 0 to 39      |

3. ACCU 2-Low - number of data words (desired block length without the header)

The maximum permissible block length is 32767 data words (including the header). Presently a programmer can handle approximately 4 Kwords.

The following operational sequence generates PB 24 with a length of 2000 data words (total length including the header is 2005 data words):

```
:L      KF2000
:L      KY1,24
:JU     OB125
```

If OB 125 is processed correctly, the RLO is set to 0. Condition codes CC 1 and CC 0 are cleared also.

In the following cases, an error aborts processing of OB 125:

- The block type or number in ACCU 1-Low is illegal.
- The number of data words (block length) in ACCU 2-Low is illegal.
- The STEP 5 block to be generated already exists in the memory.
- The memory space is insufficient to generate the entire block.

- The online programmer function COMPRESS MEMORY is running. Simultaneous processing of OB 125 is illegal.

In the event of an error the processing of a special function OB is aborted. Program processing continues with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1 (see next page).

In the following cases, a warning aborts processing of OB 125:

- An online programmer function that is in a critical area is running simultaneously (e.g., STATUS).
- OB 124, OB 125, OB 254, or OB 255 was called within the last 10 ms. (Only one of these four special functions can be called within 10 ms. This prevents the interface to the programmer from being cut off completely by multiple calls of these special function OBs.)

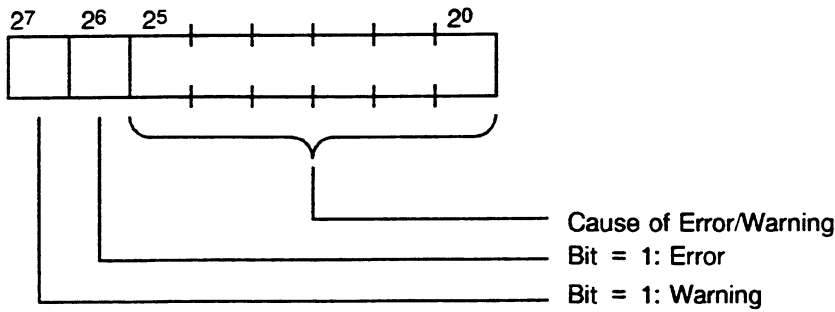
In the event of a warning, program processing is continued with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1 (see below). If a warning aborts OB 125, you can call this special function again to achieve correct processing. It may be necessary to repeat your call several times.

### Affecting the RLO and Condition Codes CC 1 and CC 0

After calling OB 125, you can use the RLO and condition codes CC 1 and CC 0 to determine whether the special function ran correctly or was aborted with an error or a warning. You can evaluate the result with conditional jump operations.

| RLO | CC 1 | CC 0 | Explanation  | Jump Operations for Evaluation |
|-----|------|------|--|--------------------------------|
| 0   | 0    | 0    | The special function was processed correctly.                  | JC                             |
| 1   | 1    | 0    | Processing of the special function was aborted with a warning. | JC, JP                         |
| 1   | 0    | 1    | Processing of the special function was aborted with an error.  | JC, JM                         |

**Location of error IDs in ACCU 1-LL for the processing of OB 125.**



**Causes of errors or warnings during the processing of OB 125:**

| Status/Error ID Numbers in ACCU 1-LL | Explanation  |
|--------------------------------------|--|
| 01H                                  | The function was processed correctly.                      |
| 42H                                  | The block already exists.                                  |
| 43H                                  | The memory space is insufficient.                          |
| 44H                                  | The block length is illegal.                               |
| 45H                                  | The block type or number is illegal.                       |
| 4DH                                  | The online programmer function COMPRESS MEMORY is running. |
| 8DH                                  | There is a conflict with another online function.          |
| 8EH                                  | The 10-ms waiting time has not elapsed.                    |

With reference to the processing of OB 125, please note the following:

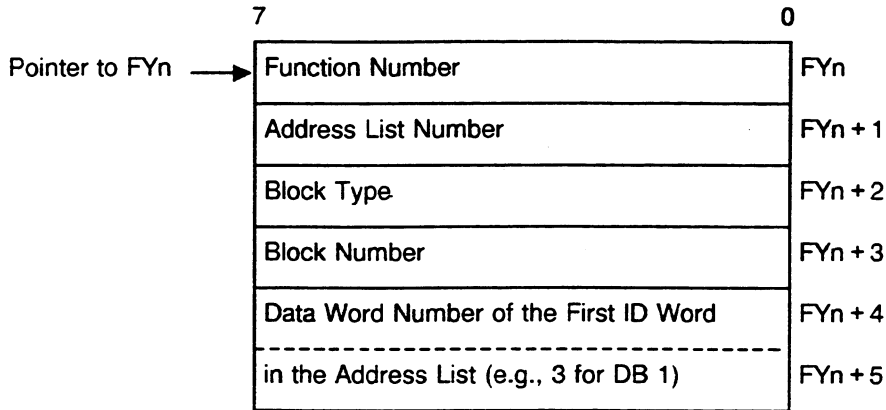
- During the actual block generation procedure, the following user interrupts are disabled: internal time interrupts, external process interrupts (150U controller mode), HW signal interrupts (155U controller mode).
- Calling OB 125 changes the contents of ACCU 1 to ACCU 4. The BR register does not change.



## 6.5 Transfer Process Image Tables (OB 126)

You can use OB 126 to define a maximum of four different process image tables. These tables can include digital inputs and outputs and input and output interprocessor communication (IPC) flags. You can define these four tables in addition to DB 1. You can read the tables into or output them from any program processing level.

To carry out the desired function, OB 126 evaluates a parameter block. You store this parameter block in the flag area of the CPU. One parameter block takes up a maximum of six flag bytes (FYs).



**Function Number (FYn):** The function number (FYn) can be 1 to 5.

Use this number to determine which function OB 126 should carry out.

- 1 : Read in the process image digital input table.
- 2 : Output the process image digital output table.
- 3 : Read in the process image input IPC flag table.
- 4 : Output the process image output IPC flag table.
- 5 : Set up the internal system address list (similar to DB 1).

(NOTE: Function 5 is permissible only during a cold restart (OB 20).)

**Address List Number (FYn + 1):** The address list number (FYn + 1) can be 1 to 4.

Note:

Setting up the entire parameter block is necessary only if you want OB 126 to generate the address list during a cold restart (function 5). To carry out functions 1 to 4, you only need to enter the function number and the address list number. The other entries do not apply.

**Block Type (FYn + 2) and Block Number (FYn + 3)**

| Block Type | Block Number |
|------------|--------------|
| 1 (DB)     | 0 to 255     |
| 2 (DX)     | 0 to 255     |

## Integrated Special Function Organization Blocks

---

### Data Word Number of the First ID Word in the Address List (FYn + 4, FYn + 5):

In flag bytes FYn + 4 and FYn + 5, enter the number of the data word that contains the first ID word in the address list. The possible ID words are:

- KH = DE00 Digital inputs
- KH = DA00 Digital outputs
- KH = CE00 Input IPC flags
- KH = CA00 Output IPC flags

For example, the first ID word in DB 1 is always DW 3. DW 0 to DW 2 contain the DB 1 start ID.

### Assign one parameter as follows:

1. ACCU 1-Low: First flag byte number (pointer to FYn) of the parameter block to be evaluated in the flag area. The permissible offset value is 0 to 250.

The offset value of the parameter must be in ACCU 1-Low before OB 126 is called.

If OB 126 is processed correctly, the RLO is set to 0. A 1 is entered in ACCU 1-LL.

### Errors:

If the special function cannot be evaluated, the processing of OB 126 is aborted. The RLO is set to 1. An error ID is indicated in ACCU 1-LL.

| Status/Error ID Numbers in ACCU1-LL | Explanation  |
|-------------------------------------|--|
| 1                                   | The function was processed correctly.  |
| 2                                   | The function number is illegal (must be 1 to 5).   |
| 3                                   | The offset in the flag area is illegal (must be 0 to 250).   |
| 4                                   | The block type or number is illegal or DB/DX does not exist.   |
| 5                                   | The first ID word is not in the specified data word of the DB/DX (wrong offset) or the address list contains an incorrect ID word. |
| 6                                   | The address list number is illegal (must be 1 to 4).   |
| 7                                   | Calling the function at this program processing level is illegal.  |

In the event of an error, processing continues with the next STEP 5 operation in the user program, with the following exception:

**Exception:** If OB 126 is supposed to carry out its function including number 5 (i.e., set up an internal system address list similar to DB 1, permissible only in OB 20), the system program checks to see if the address list is set up correctly. It checks to see if the inputs and outputs or input and output IPC flags specified in the list actually exist in the corresponding modules. If an address list has been set up incorrectly, the programmable controller acts as it would for a DB 1 error. The CPU goes into the smooth STOP mode. The "STOP" LED flashes slowly. A DB1 error is reported as the cause of the stop.

With reference to the processing of OB 126, please note the following:

- During the processing of OB 126, the following user interrupts are disabled: internal time interrupts, external process interrupts (150U controller mode), HW signal interrupts (155U controller mode)
- Calling OB 126 changes the contents of ACCU 1 to ACCU 4. The BR register does not change.
- With a warm restart, the remaining cycle is processed under "BASP." All digital outputs are disabled. At the end of the cycle, *all outputs* are reset (including those specified in address lists 1 to 4).

#### **Application Example: Setting Up an Address List and Outputting a Process Image Table**

##### **1. Setting Up an Address List in DB 5**

Set up an address list in DB 5 using programmer softkeys F1 <INPUT> and F4 <SCR FORM>. Enter "DB 5" in the BLOCK field. Program DB 5 with the following parameters:

Digital inputs: 1, 2  
Digital outputs: 3  
Input IPC flags: 5, 6, 7  
Output IPC flags: 20, 22

If you set up DB 5 manually, you must set it up like DB 1. It must have a start ID, ID words of the operand areas, and an end ID (see section 10.3.1).

##### **2. Transfer Address List in Cold Restart (OB 20)**

To transfer an address list in a cold restart using OB 20, first you must set up the parameter block in the flag area. It occupies flag bytes FY 20 to FY 25. Enter the following program:

```
:L KB 5      Transfer function number 5 (internal system address list)
:T FY 20    to flag byte 20.
:L KB 1      Transfer address list number 1
:T FY 21    to flag byte 21.
:L KH 0105  Transfer block type DB (1) and block number 5 (DB 5)
:T FW 22    to flag bytes 22 and 23.
:L KB 3      Transfer data word number 3 (DW 3 in DB 5 contains
:T FW 24    the first ID word) to flag bytes 24 and 25.
```

## *Integrated Special Function Organization Blocks*

---

After you set up the parameter block correctly, you must specify in ACCU 1-Low the first flag byte number of the parameter block. Then call OB 126. It sets up the address list. Enter the following program:

```
:L   KB 20   The parameter block begins with FY 20.  
:JU  OB 126  Call address list generation.  
:..  
:..
```

### **NOTE**

**The CPU accepts address lists with the numbers 1 to 4 only  
if OB 126 is called in OB 20 (cold restart).  
You must execute function 5 first.**

### **3. Output the Process Image Output Table**

The following STEP 5 program sequence can be at any program processing level (e.g., in OB 1, an internal time interrupt OB, or an external process interrupt OB). It outputs the process image output table for address list 1.

```
:L   KB 2     Transfer function number 2  
:T   FY 50    to flag byte 50.  
:L   KB 1     Transfer address list number 1  
:T   FY 51    to flag byte 51.  
:L   KB 50    The parameter block begins with flag byte 50.  
:JU  OB 126   Call output of the process image output table.  
:..  
:..
```

## **6.6 Multiprocessor Communication (OB 200, OB 202 to OB 205)**

You can use OB 200 and OB 202 to OB 205 to transfer data between CPUs. Such a transfer can take place in multiprocessor operation between CPUs using Coordinator 923C. These OBs are used in multiprocessing as follows:

### **OB 200 (Initialize):**

OB 200 sets up a memory in Coordinator 923C. This memory is a buffer for the data fields that are transferred.

### **OB 202 (Send):**

OB 202 transfers a data field to the buffer of Coordinator 923C and indicates how many data fields can still be sent.

### **OB 203 (Send Test):**

OB 203 determines the number of available memory blocks in the buffer of Coordinator 923C.

### **OB 204 (Receive):**

OB 204 transfers a data field from the buffer of Coordinator 923C and indicates how many data fields can still be received.

### **OB 205 (Receive Test):**

OB 205 determines the number of occupied memory blocks in the buffer of Coordinator 923C.

For a detailed user's guide to these special function organization blocks, see *Multiprocessor Communication: S5-135U, CPU 922 (R Processor) and CPU 928; S5-155U, CPU 946/947*, in the S5-135U and S5-155U manuals.

## 6.7 Compare CPU Restart Types in Multiprocessor Operation (OB 223)

In multiprocessing, use special organization block OB 223 to find out if the restart types of all participating CPUs are the same. You can use this OB during restart or at the beginning of cyclic program processing.

There are no parameters and,

consequently, no parameter-associated errors.

After OB 223 is called, you can determine the result of the test by checking the RLO and by checking the ID number indicated in ACCU 1-LL.

| RLO | Explanation  |
|-----|--|
| 0   | The restart types are the same.  |
| 1   | The restart types are not the same or there is some other type of error. |

| ID Numbers in ACCU 1-LL | Explanation  |
|-------------------------|--|
| 01H                     | The restart types are the same.  |
| 02H                     | There was an internal system error.  |
| 03H                     | The restart types are not the same.  |
| 04H                     | The programmable controller is set to single processor operation: a comparison of restart types is not possible. |

Independent of the type of restart set, the CPU in which OB 223 was called continues program processing with the next operation in the user program.

Note:

Calling OB 223 changes the contents of ACCU 1 to ACCU 4. The BR register does not change.

## 6.8 Copy/Duplicate Data Blocks (OB 254, OB 255)

You can use the special function blocks OB 254 and OB 255 to transfer data blocks in the user memory of the CPU. OB 254 and OB 255 run identically. OB254 is exclusively for extended data blocks (DXs) and OB 255 is exclusively for regular data blocks (DBs). When OB 254 or OB 255 is called, the destination data blocks must not already exist.

When transferring data blocks, you can copy them or duplicate them:

- **Copying a Data Block from the EPROM to the RAM**

A data block in the EPROM is transferred to the RAM, keeping its original block number. The new start address of the data block is entered in the address list in DB 0. The old address of the block is overwritten in DB 0 (i.e., the original data block in the EPROM is declared invalid). If the new data block in the RAM is deleted later, the original data block is valid in the EPROM again after a cold restart.

Assign the following two parameters:

1. ACCU 1-LL - number of the data block to be copied
2. ACCU 1-LH - 0

Permissible Block Numbers for OB 254 and OB 255

| Block Type  | Block Number |
|-------------|--------------|
| DB (OB 255) | 1 to 255     |
| DX (OB 254) | 0 to 255     |

The following operational sequence copies DX 120 from the EPROM to the RAM. DX 120 in the EPROM is declared invalid.

```
:L    KY 0,120
:JU   OB 254
```

If OB 254/OB 255 is processed correctly, the RLO is set to 0. Condition codes CC 1 and CC 0 are also cleared.

The following errors are possible:

- The data block to be copied does not exist in the EPROM.
- The block already exists in the RAM.
- The block number in ACCU 1-LL is illegal.
- The memory space is insufficient.
- The online programmer function COMPRESS MEMORY is running. Simultaneous processing of OB 254/OB 255 is illegal.

### • Duplicating a Data Block in the RAM

A data block is transferred in the RAM and receives a new 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 in DB 0 is maintained (i.e., the original data block is still valid).

The start address is not entered in DB 0 until the transmission is completed and all IDs in the block header are entered correctly. Therefore, the duplicated block is not recognized as valid or existing until after the transmission is complete.

Assign the following two parameters:

1. ACCU 1-LL - number of the data block to be duplicated (source)
2. ACCU 1-LH - number of the new block (destination)

Permissible block numbers for OB 254 and OB 255.

| Block Type  | Block Number |
|-------------|--------------|
| DB (OB 255) | 1 to 255     |
| DX (OB 254) | 0 to 255     |

The following operational sequence duplicates DB 80. The number of the new data block is 85. The contents of DB 80 and DB 85 are identical.

```
:L    KY 85,80  
:JU   OB 255
```

If OB 254/OB 255 is processed correctly, the RLO is set to 0. Condition codes CC 1 and CC 0 are also cleared.

The following errors are possible:

- The data block to be duplicated (source) does not exist in the RAM.
- The new data block (destination) already exists in the RAM.
- The block number in ACCU 1-LL or ACCU 1-LH is illegal .
- The memory space is insufficient.
- The online programmer function COMPRESS MEMORY is running. Simultaneous processing of OB 254/OB 255 is illegal.

An error aborts the processing of the special function organization block for the duplicating function. Program processing is continued with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1 (see next page).



In the following cases, a warning aborts the processing of OB 254/OB 255:

- An online programmer function that is in a critical area is running simultaneously (e.g., STATUS).
- OB 124, OB 125, OB 254, or OB 255 was called within the last 10 ms. (Only one of these four special functions can be called within 10 ms. This prevents the interface to the programmer from being cut off completely by multiple calls of these special function OBs.)

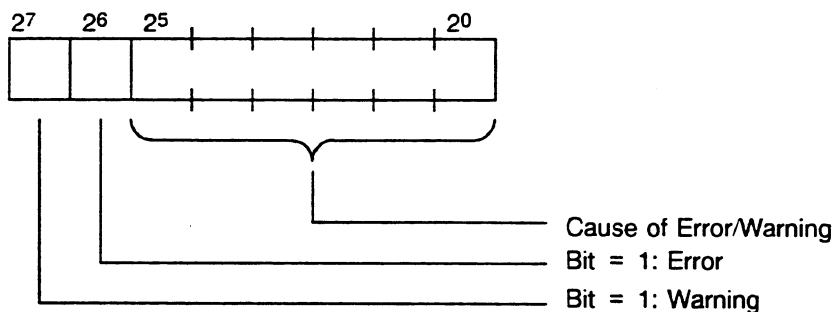
In the event of a warning, program processing is continued with the next STEP 5 operation. The RLO is set to 1. An error ID is indicated in ACCU 1 (see below). If OB 254/OB 255 is aborted with a warning, you can call this special function again to achieve correct processing. It may be necessary to repeat your call several times.

**Affecting the RLO and Condition Codes CC 1 and CC 0**

After calling OB 254/OB 255, you can use the RLO and condition codes CC 1 and CC 0 to determine whether the special function ran correctly or was aborted with an error or a warning. You can evaluate the result with conditional jump operations.

| RLO | CC 1 | CC 0 | Explanation  | Jump Operations for Evaluation |
|-----|------|------|--|--------------------------------|
| 0   | 0    | 0    | The special function was processed correctly.                  | JC                             |
| 1   | 1    | 0    | Processing of the special function was aborted with a warning. | JC, JP                         |
| 1   | 0    | 1    | Processing of the special function was aborted with an error.  | JC, JM                         |

**Location of error IDs in ACCU 1-LL for processing of OB 254/OB 255.**



## *Integrated Special Function Organization Blocks*

---

Causes of errors or warnings during the processing of OB 254/OB 255:

| Status/Error ID<br>Numbers in<br>ACCU 1-LL | Explanation   |
|--|---|
| 01H  | The function was processed correctly.                         |
| 48H  | The source data block does not exist.                         |
| 49H  | The memory space is insufficient.                             |
| 4AH  | The block number (source) is illegal.                         |
| 4BH  | The block number (destination) is illegal.                    |
| 4CH  | The destination data block already exists in the RAM.         |
| 4DH  | The online programmer function COMPRESS<br>MEMORY is running. |
| 8DH  | There is a conflict with another online function.             |
| 8EH  | The 10-ms waiting time has not elapsed.                       |

With reference to the processing of OB 254/OB 255, please note the following:

- During the actual transmission procedure, the following user interrupts are disabled: internal time interrupts, external process interrupts (150U controller mode), HW signal interrupts (155U controller mode)
- Calling OB 254/OB 255 changes the contents of ACCU 1 to ACCU 4. The BR register does not change.

## Chapter 7

### Extended Data Block DX 0

You can set system program presettings for your specific requirements in extended data block DX 0 (see table on page 7-4 for the default presettings, designated by the arrow symbol "→").

The default presettings of the system program are set automatically during every cold restart. DX 0 is evaluated afterwards. If you do not program DX 0, the default settings apply; otherwise, your settings apply.

#### NOTE

**Entries or changes to DX 0 become effective only during a cold restart.**

The parameters in DX 0 can influence the following operations of the CPU:

- mode of operation
- start-up program processing in single processor operation or with multiprocessing
- cyclic program processing
- time driven program processing
- interrupt driven program processing

**Note:** DX 0 can be assigned parameters either via the appropriate screen form in the PG software or directly in hexadecimal form.

# Extended Data Block DX 0

## Structure of DX 0

DX 0 is made up of the following three parts:

1. start ID for DX 0 (DW 0, DW 1, and DW 2)
2. several blocks of varying lengths (depending on the number of parameters)
3. end delimiter EEEE

The numbers specified are in hexadecimal format.

Structure of DX 0:

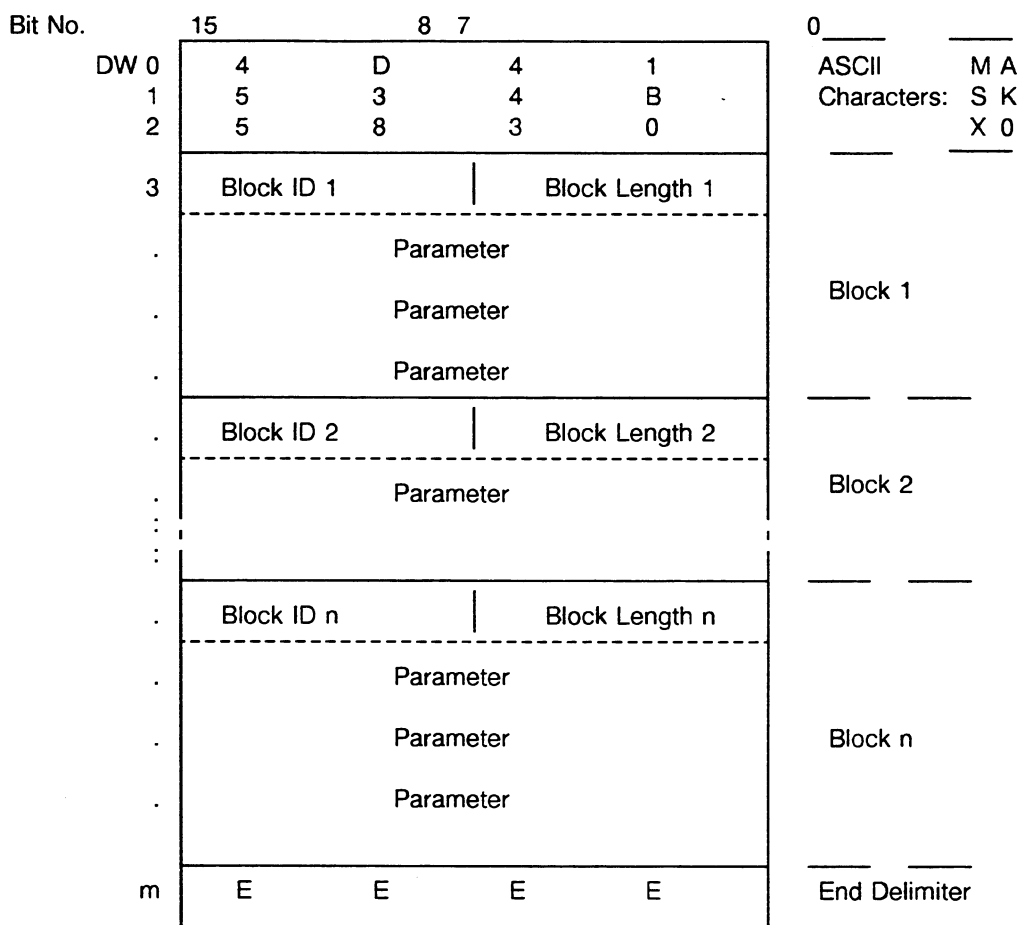


Fig. 7.1 Structure of DX 0

**Example: Entering DX 0**

|  |   |
|--|---|
| Start ID   | DW 0: KH = 4D41<br>DW 1: KH = 534B<br>DW 2: KH = 5830         |
| Block ID/length<br>Parameter (takes up one DW)   | DW 3: KH = 0101<br>DW 4: KH = 1001 Block 1                    |
| Block ID/length<br>Parameters (takes up two DWs) | DW 5: KH = 0402<br>DW 6: KH = 1000 Block 2<br>DW 7: KH = 0040 |
| End delimiter                                    | DW 8: KH = EEEE   |

A **block** in DX 0 consists of 1 to n data words.

These data words contain the following:

- block ID
- block length
- block parameter

The **block ID** explains the parameters that follow the block ID. Each block is assigned to a specific system program part or to a specific system function (e.g., block ID 04 is assigned to cyclic program processing).

The **block length** indicates the number of data words needed for the parameters that follow the block length indication.

The possible **parameters** are described on the following pages.

When assigning parameters in DX 0, note the following:

- You do not need to specify blocks that you are not going to use.
- You must enter the blocks in sequence (e.g., enter a block with the ID 02 before you enter a block with ID 05).
- A specific block can occur only once in DX 0.
- If you specify a block, the number of parameters must agree with the given block length.
- You must follow the parameter sequence.

**NOTE**

**After you enter the last block, you must terminate DX 0 with the end delimiter "EEEE."**

Possible Specifications in DX 0

| Block ID/Length  | Parameter 1st Word/ 2nd Word | Explanation <sup>1)</sup>   |
|--|------------------------------|---|
| <b>Controller Modes of Operation</b>                   |                              |   |
| 01xx <sup>2)</sup>                                     | 1000                         | → 150U controller mode  |
|  | 1001                         | 155U controller mode (see Section 1.4)  |
| <b>Start-Up Program Processing</b>                     |                              |   |
| 02xx   | 1000<br>1001<br>1002         | → Automatic warm restart after power up<br>Automatic cold restart after power up<br>Manual cold restart/warm restart after power up   |
|  | 2000                         | → Synchronization of start-up in multiprocessing operation  |
|  | 2001                         | No synchronization of start-up in multiprocessing operation   |
|  | BB00 00yy <sup>3)</sup>      | Number of timers (yy) allowed to be updated (hexadecimal format): $0 \leq yy \leq FF$ <sup>4)</sup><br>→ yy = FF (timer T0 to T255)   |
|  | 4000<br>4001                 | → Warm restart type: warm restart<br>Warm restart type: retentive cold restart <sup>5)</sup>  |
| <b>Cyclic Program Processing</b>                       |                              |   |
| 04xx   | 1000 00yy                    | Setting the cycle time <sup>6)</sup><br>Cycle time = (yy x 10 ms) permissible values (hexadec. format): $1 \leq yy \leq FF$<br>→ yy = 14 (200 ms)   |
|  | 4000<br>4001                 | → Update of the process image of the interprocessor communication flags without semaphore protection<br><br>Update of the process image of the interprocessor communication flags with semaphore protection (in groups of 32 bytes, see section 10.2) |
| <b>Interrupt Processing (Internal Time Interrupts)</b> |                              |   |
| 05xx   | 1000 000c<br>1001 0000       | → Internal time interrupt processing on<br>Internal time interrupt processing off<br>c := level priority 1 to 5<br>→ c := level priority 1 (highest priority)   |
|  | 2000 00yy                    | Basic clock rate for internal time interrupt processing<br>Basic clock rate = (yy x 10 ms) permissible values (hexadecimal format): $1 \leq yy \leq FF$<br>→ yy = 0A (100 ms)   |
|  | 3000<br>3001                 | → Clock distributor like the S5-150U<br>Clock distributor 2 <sup>n</sup>  |

| Block ID/Length   | Parameter 1st Word/ 2ndWord  | Explanation  |
|---|------------------------------|--|
| Interrupt Processing (HW Signal Interrupts via the S5 Bus)/155U Controller Mode |                              |  |
| 05xx  | 4000    000c<br>4001    0002 | System interrupt A/B on<br>→ System interrupt A/B off  |
|   | 5000    000c<br>5001    0002 | System interrupt E on<br>→ System interrupt E off  |
|   | 6000    000c<br>6001    0002 | System interrupt F on<br>→ System interrupt F off  |
|   | 7000    000c<br>7001    0002 | System interrupt G on<br>→ System interrupt G off  |
|   | 8000    000c<br>8001    0000 | → Process interrupt on<br>Process interrupt off<br>c := Level priority 1 or 2<br>→ c := Level priority 2 |
| EEEE  |                              | End delimiter  |

- 1) Under the heading "Explanation," → indicates the default presetting, if DX 0 is not loaded or missing.
- 2) xx stands for the block length (number of data words taken up by the parameters).
- 3) Parameters that take up two data words must be counted as two data words when you calculate the block length.
- 4) Note the following concerning **timer update**:
  - Normally timers T 0 to T 255 are updated.
  - If you enter the number 0 in DX 0, no timer is updated, not even if it is contained in the program. An error message is not given.
  - Timers are updated in *pairs* as follows:

|         |      |            |            |            |            |     |
|---------|------|------------|------------|------------|------------|-----|
| Entry:  | '0'  | '1' or '2' | '3' or '4' | '5' or '6' | '7' or '8' | ... |
| Update: | None | T0 or T1   | T0 to T3   | T0 to T5   | T0 to T7   | ... |

- 5) The retentive cold restart maintains the signal status of the flags and the address list (DB 0).
- 6) You can also set the cycle time in OB 31. It is called during every cold restart. Load a value between 1 and 255 into ACCU 1. The set cycle time is then (ACCU 1) x 10 ms.

**Note:** If OB 31 and DX 0 contain different specifications for the cycle time, the values in OB 31 take precedence.

**Interrupt Level Priorities**

The 150U and 155U controller modes have varying numbers of interrupt levels that you can prioritize as follows:

**150U controller mode (default presetting):**

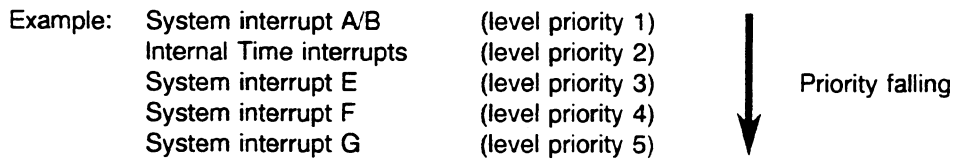
- a) Internal time interrupts:
  - the default presetting is level priority 1 (higher priority).
- b) External process interrupts via input byte "IB 0":
  - the default presetting is level priority 2 (lower priority).

You can switch the level priorities in DX 0 for internal time interrupts and external process interrupts in the 150U controller mode.

**155U controller mode (must be set in DX 0):**

- a) internal time interrupts
- b) system interrupt A/B
- c) system interrupt E
- d) system interrupt F
- e) system interrupt G

You can prioritize these five interrupt levels in DX 0 as follows: level 1 is the highest priority.





**Assigning Parameters in DX 0 Using the DX 0 Screen on a Programmer**

| D X 0 - P A R A M. ASS. (S5-155U)    |     |   |     |     |                 |     |     |
|--------------------------------------|-----|---|-----|-----|-----------------|-----|-----|
| MODE:                                |     | <b>150U</b>   |     |     |                 |     |     |
| RESTART AFTER POWER UP :             |     | <b>1</b> ( 1 = WARM RESTART<br>2 = COLD RESTART<br>3 = MANUAL START ) |     |     |                 |     |     |
| WARM RESTART PROCEDURE:              |     | <b>1</b> ( 1 = WARM RESTART<br>2 = COLD RESTART WITH MEMORY )         |     |     |                 |     |     |
| NUMBER OF TIMER CELLS:               |     | <b>256</b> ( 0...256 )  |     |     |                 |     |     |
| CYCLE TIME MONITORING (X 10 MS) :    |     | <b>20</b> ( 1...255 )   |     |     |                 |     |     |
| SYNCHRONIZE MULTIPROCESSOR RESTART : |     | <b>YES</b>  |     |     |                 |     |     |
| BLOCK TRANSFER OF THE IPC FLAGS :    |     | <b>NO</b>   |     |     |                 |     |     |
| F 1                                  | F 2 | F 3<br>SELECT   | F 4 | F 5 | F 6<br>CONTINUE | F 7 | F 8 |

The **parameters** in bold typeface in the input fields correspond to the default presettings of the S5-155U.

| D X 0 - P A R A M. ASS. (S5-155U)              |     |               |     |   |                 |     |     |
|--|-----|---------------|-----|---|-----------------|-----|-----|
| TIME INT.:                                     |     |               |     |   |                 |     |     |
| TIME INT. SERVICING :                          |     | <b>YES</b>    |     | PRIORITY : 1  |                 |     |     |
| BASIC CLOCK (X 10 MS) :                        |     | <b>10</b>     |     | ( 1...255 )   |                 |     |     |
| CLOCK PULSE PROCESS. :                         |     | <b>1</b>      |     | ( 1 = FACTOR 1, 2, 5, 10<br>2 = FACTOR 1, 2, 4, 8 ) |                 |     |     |
| HARDWARE PROCESS INT. (ONLY IN 155U MODE):     |     |               |     |   |                 |     |     |
| SYSTEM INTERRUPT A/B :                         |     | <b>NO</b>     |     | PRIORITY : 2  |                 |     |     |
| SYSTEM INTERRUPT E :                           |     | <b>NO</b>     |     | PRIORITY : 2  |                 |     |     |
| SYSTEM INTERRUPT F :                           |     | <b>NO</b>     |     | PRIORITY : 2  |                 |     |     |
| SYSTEM INTERRUPT G :                           |     | <b>NO</b>     |     | PRIORITY : 2  |                 |     |     |
| PROCESS INT. INPUT BYTE 0 (ONLY IN 150U MODE): |     |               |     |   |                 |     |     |
| PROCESS INT. :                                 |     | <b>YES</b>    |     | PRIORITY : 2  |                 |     |     |
| F 1  | F 2 | F 3<br>SELECT | F 4 | F 5   | F 6<br>CONTINUE | F 7 | F 8 |

### Assigning Parameters in DX 0 Manually

KH = 4D41; Start ID M A  
KH = 534B; S K  
KH = 5830; X 0

KH = 0101; Block ID, block length (= 1 word)  
KH = 1000; 150U controller mode

KH = 0205; Block ID, block length (= 5 words)  
KH = 1000; Automatic warm restart after power up  
KH = 2000; Synchronization of start-up in multiprocessor operation  
KH = BB00; Timer update  
KH = 00FF; T 0 to T 255  
KH = 4000; Type of warm restart: warm restart

KH = 0403; Block ID, block length (= 3 words)  
KH = 1000; Cycle monitoring time  
KH = 0014; 200 ms (20 x 10 ms)  
KH = 4000; Update of the process image of the IPC flags without semaphore protection

KH = 050F; Block ID, block length (= 15 words)  
KH = 1000; Internal time interrupt processing on,  
KH = 0001; priority 1  
KH = 2000; Basic clock rate for internal time interrupt processing,  
KH = 000A; 100 ms (10 x 10 ms)  
KH = 3000; Clock distributor like the S5-150U (1, 2, 5, 10 ...)  
KH = 4001; System interrupt A/B off,  
KH = 0002; priority 2  
KH = 5001; System interrupt E off,  
KH = 0002; priority 2  
KH = 6001; System interrupt F off,  
KH = 0002; priority 2  
KH = 7001; System interrupt G off,  
KH = 0002; priority 2  
KH = 8000; External process interrupts on,  
KH = 0002; priority 2

KH = EEEE; End delimiter

The parameters shown above are the default presettings of CPU 946/947. You can change them to suit your programming needs or you can eliminate blocks that you do not need. When you load DX 0 into the user memory, the block is valid after the next cold restart.

**If a new DX 0 is loaded after a cold restart, the existing parameters remain unchanged.**

**Parameter Assignments in DX 0**

You want to use three CPUs in multiprocessor operation: CPU A, CPU B, and CPU C. CPU A and CPU B work together closely, exchange data frequently, and each processes an extensive start-up program. CPU C works for the most part independently, processing a short, time-critical program.

Normally, all CPUs begin cyclic program processing at the same time in multiprocessor operation (i.e., the CPUs wait for each other to end their start-up phase and then begin cyclic program processing together).

Since CPU C executes its program independently and its start-up program is very short, start-up synchronization is not necessary for CPU C. You can assign parameters in DX 0 so that CPU C goes into cyclic program processing immediately after it finishes its start-up phase, without having to wait for CPU A and CPU B.

Assign parameters in DX0 as shown. You must specify all parameters belonging to a block.

|           |                       |       |           |
|-----------|-----------------------|-------|-----------|
| <u>DX</u> | Start ID              | DW 0: | KH = 4D41 |
|           |                       | DW 1: | KH = 534B |
|           |                       | DW 2: | KH = 5830 |
|           | First Block ID/Length | DW 3: | KH = 0205 |
|           | Parameter 1           | DW 4: | KH = 1000 |
|           | Parameter 2           | DW 5: | KH = 2001 |
|           | Parameter 3           | DW 6: | KH = BB00 |
|           | Parameter 4           | DW 7: | KH = 00FF |
|           | Parameter 5           | DW 8: | KH = 4000 |
|           | End delimiter         | DW 9: | KH = EEEE |

When you load this DX 0 into the program memory, it takes effect after the next cold restart. Since CPU C has a very short start-up program and does not have to wait for CPU A and CPU B, its green "RUN" LED lights up immediately. However, the disable output command signal ("BASP") is not canceled until all three CPUs have ended their start-up phase and go into the RUN mode. This means that CPU C cannot access the digital I/Os until that time.

## Chapter 8

### Memory Assignment and Memory Organization

The entire memory area of CPU 946/947 is divided into the following areas.

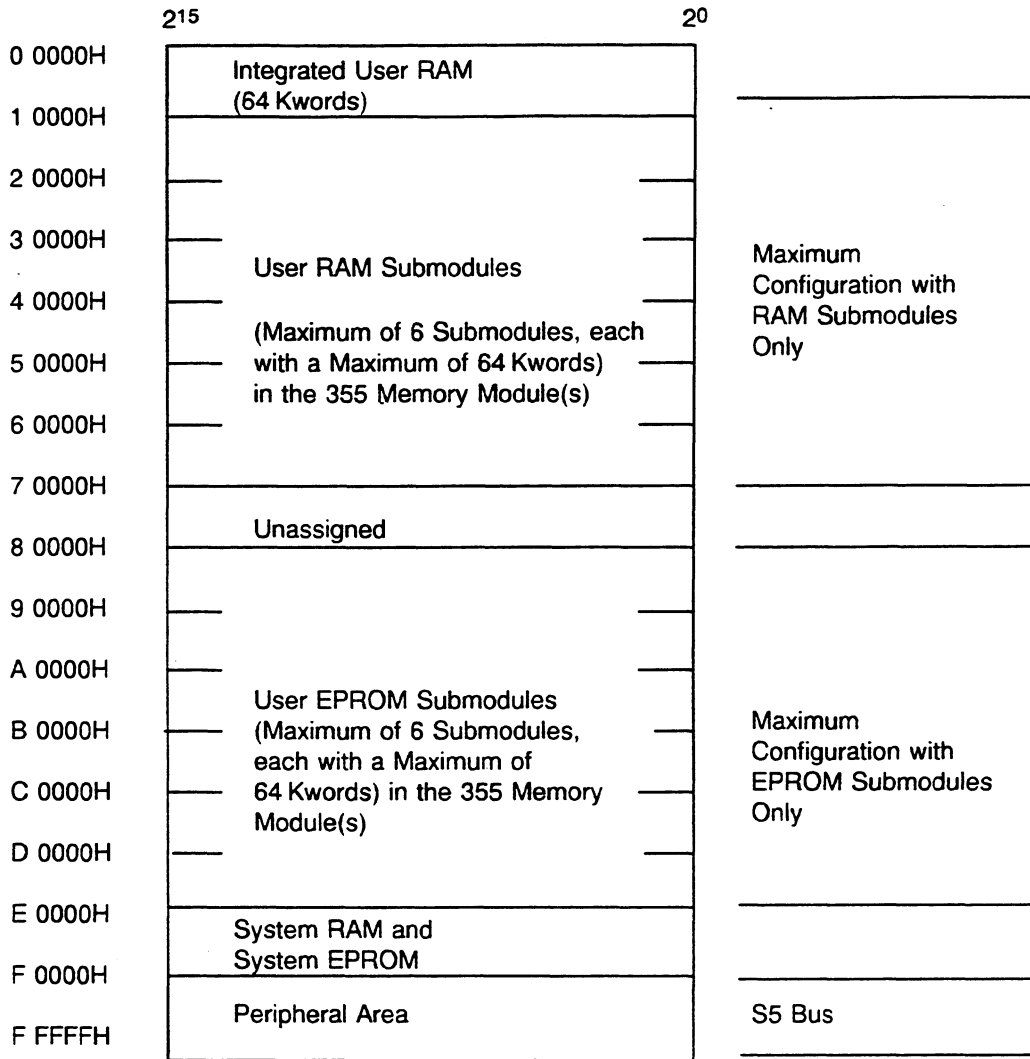
|   | Data Width: |                  |
|---|-------------|------------------|
|   | -----       | -----            |
| 1. User memory<br>for OBs, FBs, FXs, PBs, SBs, DBs, DXs | 16 bits     |                  |
| 2. - Serial communications interface area: RI, RJ       | 16 bits     | CPU<br>internal  |
| - System area: RS, RT                                   | 16 bits     |                  |
| - Counters: C   | 16 bits     |                  |
| - Timers: T   | 16 bits     |                  |
| 3. - Flags: F   | 8 bits      |                  |
| - Flags: S  | 8 bits      |                  |
| - Process image input/output tables: PII, PIQ           | 8 bits      |                  |
|   |             | -----            |
| 4. Peripheral area:                                     |             |                  |
| - "P" peripherals                                       | 8 bits      |                  |
| - "O" peripherals                                       | 8 bits      |                  |
| - Interprocessor communication flags                    | 8 bits      | On the<br>S5 bus |
| - Coordinator (COOR)/global memory (semaphore...)       | 8 bits      |                  |
| - Dual-port RAM pages (CP, IP, COOR 923 C)              | 8/16 bits   |                  |
| - Distributed peripherals                               | 8 bits      |                  |
| - Hardware registers                                    | 8/16 bits   |                  |
|   |             | -----            |

Figure 8.1 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.**

8.1 Memory Assignment in CPU 946/947



The maximum submodule configuration (RAM and/or EPROM) is six, each with a maximum of 64 Kwords.

Fig. 8.1 CPU 946/947 Memory Map

8.1.1 Memory Assignment for the System RAM

|         | 215                         | 27 | 20      |
|---------|-----------------------------|----|---------|
| E 8200H | System Program Data         |    |         |
| E 9FFFH |                             |    |         |
| E A000H |                             |    | S flags |
| E AFFFH |                             |    |         |
| E B000H | System Program Data         |    |         |
| E DEAFH |                             |    |         |
| E DEB1H | BSTACK (40 Entries)         |    |         |
| E DF6FH | Reserved                    |    |         |
| E DFA0H | ISTACK (Entry No. 1)        |    |         |
| E DFA1H | ISTACK (16 Further Entries) |    |         |
| E E1F0H | Reserved                    |    |         |
| E E1FBH | DB 0 Header                 |    |         |
| E E200H | Address List OB 0 to OB 255 |    |         |
|         | Reserved                    |    |         |
| E E400H | Address List PB 0 to PB 255 |    |         |
|         | Reserved                    |    |         |
| E E600H | Address List SB 0 to SB 255 |    |         |
|         | Reserved                    |    |         |
| E E800H | Address List FB 0 to FB 255 |    |         |
|         | Reserved                    |    |         |
| E EA00H | Address List FX 0 to FX 255 |    |         |
|         | Reserved                    |    |         |
| E EC00H | Address List DB 0 to DB 255 |    |         |
|         | Reserved                    |    |         |
| E EE00H | Address List DX 0 to DX 255 |    |         |
|         | Reserved                    |    |         |

DB 0  
 (Contains the Paragraph Addresses of all Blocks {i.e., Address Bits 2<sup>4</sup> to 2<sup>19</sup>})

Memory Assignment and Memory Organization

|         |   |          |
|---------|---|----------|
|         | 215   | 20       |
| E F000H | RS Area (System Data, 256 Words)                              |          |
|         | Reserved  |          |
| E F200H | RT Area (Extended System Data, 256 Words)                     |          |
|         | Reserved  |          |
| E F400H | RI Area (Serial Communications Interface, 256 Words)          |          |
|         | Reserved  |          |
| E F600H | RJ Area (Extended Serial Communications Interface, 256 Words) |          |
|         | Reserved  |          |
| E F800H | Counters (256)  |          |
|         | Reserved  |          |
| E FA00H | Timers (256)  |          |
|         | Reserved  |          |
| E FC00H |   | Flags    |
| E FD00H |   | Reserved |
| E FE00H |   | PII      |
| E FE80H |   | PIQ      |
| E FF00H |   | Reserved |
| E FF80H |   | Reserved |
| E FFFFH |   | Reserved |
|         | 27  | 20       |

Fig. 8.2 System RAM Memory Map

8.1.2 Memory Assignment for the Peripherals

|         |  |  |    |        |
|---------|--|--|----|--------|
|         | 215  | 28 27  | 20 |        |
| F 0000H | Unassigned Peripheral Address Space<br>(52 Kwords)                                       |  |    |        |
| F D000H | Reserved   |  |    |        |
| F F000H |  | Digital I/Os<br>(with PI)<br>(128 I/128 Q)   |    | P area |
| F F080H |  | Analog I/Os<br>(without PI)<br>(128 I/128 Q)   |    |        |
| F F100H |  | Extended<br>Peripherals<br>(Only in Expansion<br>Units)  |    | O area |
| F F200H |  | Interprocessor<br>Communication Flags<br>in COOR and/or CP   |    |        |
| F F300H |  | Semaphores (32)<br>in COOR   |    |        |
| F F400H | Data Transfer Area for CPs<br>(Dual-Port RAM Pages, Each 1 Kword Long)                   |  |    |        |
| F F800H | Additional Data Area for CPs<br>(Extended Dual-Port RAM Pages, Total<br>Length 2 Kwords) |  |    |        |
| F FC00H |  | Distributed Periph-<br>erals, Extended Address<br>Set with IM 302, IM 304,<br>IM 307 and IM 308<br>Interface Modules |    |        |
| F FE00H | HW Registers   |  |    |        |
| F FFFFH |  |  |    |        |

Fig. 8.3 Address Areas for Periphery (8 bits)



Address Areas for Peripherals/Programming

| Absolute Address of Area   | Referenced with:   | Parameter  |
|--|--|--|
| <p>E FE00</p> <div style="border: 1px solid black; padding: 2px; width: fit-content;">                     PII<br/>(Process image input table)                 </div> <p>E FE7F</p><br><p>E FE80</p> <div style="border: 1px solid black; padding: 2px; width: fit-content;">                     PIQ<br/>(Process image output table)                 </div> <p>E FEFF</p><br><p>F F000</p> <div style="border: 1px solid black; padding: 2px; width: fit-content;">                     Digital peripherals,<br/>Inputs/outputs                 </div> <p>F F07F</p> <p>"P" peripherals with process image</p> | <p>L IB/T IB<br/>L IW/T IW<br/>L ID/T ID</p><br><p>A I/AN I/O I/ON I<br/>S I/R I/ = I</p><br><p>L QB/T QB<br/>L QW/T QW<br/>L QD/T QD</p><br><p>A Q/AN Q/O Q/ON Q<br/>S Q/R Q/ = Q</p><br><p>L PY/T PY<br/>L PW/T PW</p> | <p>0 to 127<br/>0 to 126<br/>0 to 124<br/>0.0 to 127.7</p><br><p>0 to 127<br/>0 to 126<br/>0 to 124<br/>0.0 to 127.7</p><br><p>0 to 127<br/>0 to 126</p> |
| <p>F F080</p> <div style="border: 1px solid black; padding: 2px; width: fit-content;">                     Digital or analog peripherals<br/>Inputs/outputs                 </div> <p>F F0FF</p> <p>"P" peripherals without process image</p>  | <p>L PY/T PY</p><br><p>L PW/T PW</p>   | <p>128 to 255</p><br><p>128 to 254</p>   |
| <p>F F100</p> <div style="border: 1px solid black; padding: 2px; width: fit-content;">                     Extended peripherals<br/>Inputs/outputs                 </div> <p>F F1FF</p> <p>"O" peripherals</p>   | <p>L OB/T OB</p><br><p>L OW/T OW</p>   | <p>0 to 255</p><br><p>0 to 254</p>   |

Fig. 8.4 Address Areas for Peripherals/Programming

### Accessing Peripherals

Using STEP 5 operations, you can access peripherals 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.

You can use the following operations to access peripherals directly :

L PY, T PY,  
L PW, T PW,  
L OB, T OB,  
L OW, T OW

When these operations are processed, the CPU reads or sets the inputs and outputs. The system program updates the outputs in the process image for digital peripherals (0 to 127).

You can use the following operations to access peripherals via the process image:

L IB, T IB, L IW, T IW, L ID, T ID, L QB, T QB,  
L QW, T QW, L QD, T QD, A I, AN I, O I, ON I, S I,  
R I, = I, A Q, AN Q, O Q, ON Q, S Q, R Q, = Q

When these operations are processed, the system program updates only the process image. The system program transfers the new, total status of the process image to the peripherals only at the end of the cycle.

When using peripherals in the O area, please refer to the appropriate operations listed in section 3.2 of this book and information in the S5-155U Central Controller Housing Hardware and Installation Guide.

#### Note:

Using your program you can access distributed address areas via the interface modules IM 302, IM 304, IM 307 or IM 308. This provides you with two new address areas which are equivalent to the O area. However, in contrast to the O area, you can only access these areas via absolute addressing.

### 8.2 Memory Organization in CPU 946/947

The user memory occupies the memory area from "00000H" to "DFFFFH." When you load the individual blocks of your program, they are stored in the memory in random order (with addresses in ascending order).

When you correct a block, the old block in the memory is marked as invalid (i.e., the start ID is overwritten) and a new block is entered in the memory and the address list. This procedure also applies when you delete a block. However, the old block in the memory is not really deleted. Its block ID/block number is overwritten with "3FFFH," declaring it invalid and erasing it from the address list. Gaps created by deletion are managed as available memory locations and are used again when you load new blocks.

The COMPRESS MEMORY programmer 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. Consequently, large available memory areas result. You can use these areas for loading new blocks.

If COMPRESS MEMORY 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 CPU 946/947, blocks are stored such 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 "xxxx0H" (bits 2<sup>0</sup> to 2<sup>3</sup> = 0). The gaps that result between blocks are filled in by filler blocks, so that all blocks continue to exist in consecutive order.

Memory management handles filler blocks like the other blocks:

|                          |         |                             |
|--------------------------|---------|-----------------------------|
| Start ID:                | 70 70 H | ;                           |
| Block type/block number: | 01 FB H | ; stands for a filler block |
| Programmer ID:           | 00 FF H | ; irrelevant                |
| Library number:          | FF FF H | ; irrelevant                |
| Block length:            | 00 XY H | ; length 5 to 20 words      |
| Data:                    | FF FF H | ; according                 |
|                          | :       | ; to                        |
|                          | :       | ; length;                   |
|                          | FF FF H | ; can be left out entirely  |

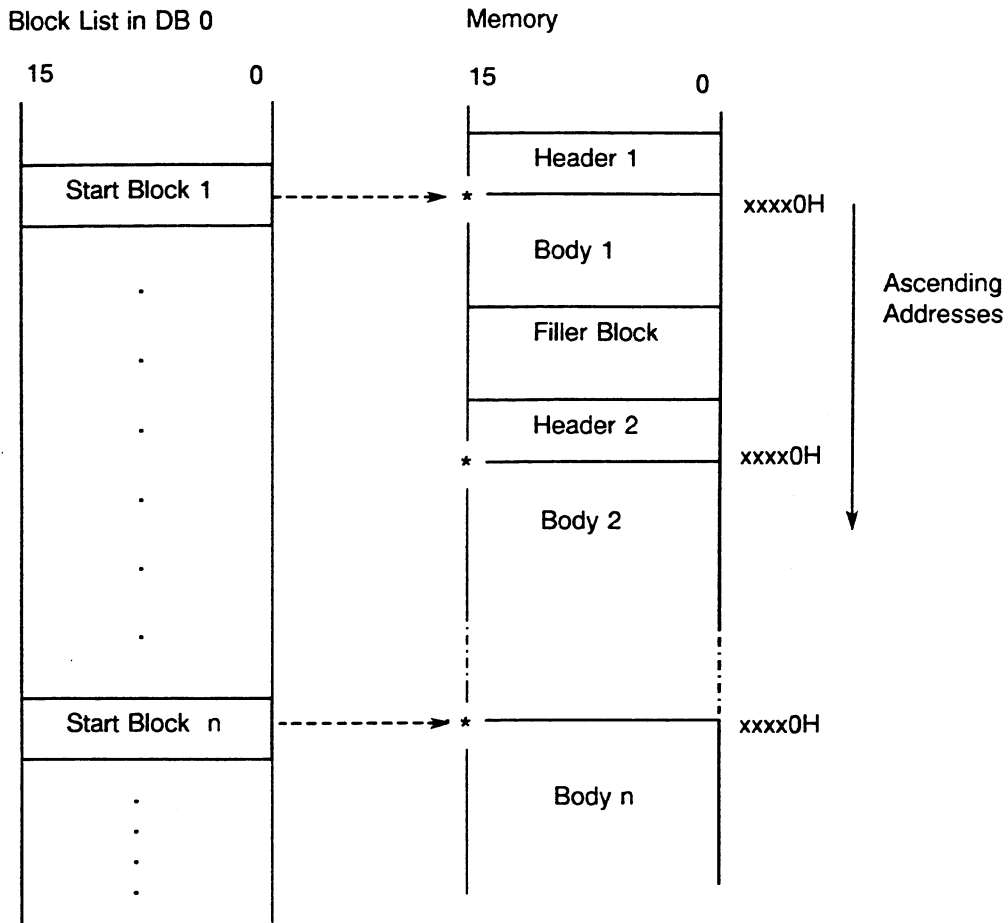


Fig. 8.5 Location of Blocks in Memory

\* Indicates a paragraph addresses (16-word boundary)

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:

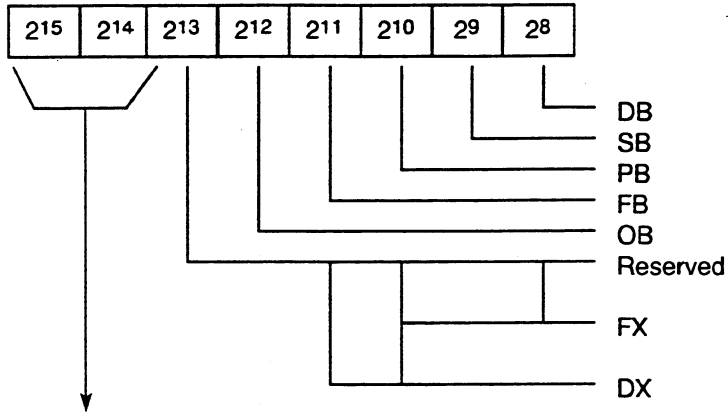
| Difference | Calculation for Length of Filler Block (Including Header) |
|------------|---|
| 0 to 5     | Add 10 to the difference                                  |
| 6          | No filler block is inserted                               |
| 7 to 10    | Add 10 to the difference                                  |
| 11 to 15   | Subtract 6 from the difference                            |

### 8.2.1 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: This word contains the block start ID, 7070H.

2nd word: The high byte of this word contains the block type.



- 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.
- 1 0 The block in the RAM is valid; it can be used in the EPROM.
- 1 1 The block in the EPROM is valid.

Low byte = block number

The low byte 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.

### 8.2.2 Block Address List in Data Block DB 0

Data block DB 0 is located in the system RAM of the CPU (beginning at address "EE200H"). 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.

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.

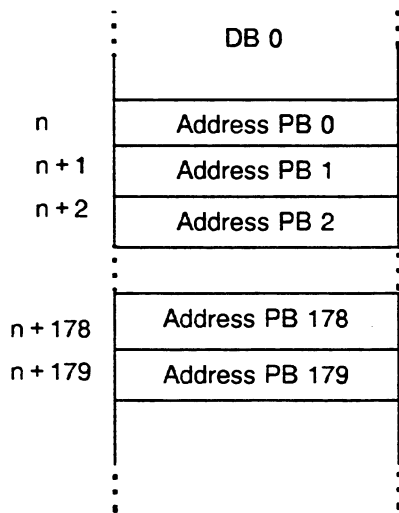
The start addresses of each block address list are specified. The following example explains how to determine the start address (DW 0 or the first STEP 5 statement) of a block.

Example: The list of block start addresses for program blocks (PBs) begins with address "EE400H." Consequently, you can read out the start address of PB 22 using a memory access to the address "EE416H."

The start addresses always point to the **data word DW 0** (or the first STEP 5 operation). Because each block lies at a paragraph address (16-bit boundary), each entry in the address list in DB 0 is limited to 1 word with bits 2<sup>4</sup> to 2<sup>20</sup> of the address.

#### Location of block addresses in DB 0.

n = EE400H (start address of the PB address list)



If 0 is entered as the address, the block is not loaded.

### **8.2.3 RI/RJ Area**

The RI and RJ areas are areas that are 256 words long in the internal system RAM of the CPU. RI occupies addresses "E F400H" to "E F4FFH."

RJ occupies addresses "E F600H" to "E F6FFH."

You can use the entire RI area (RI0 to RI255) and the entire RJ area (RJ0 to RJ255) for your own purposes.

Only an overall reset can clear the RI/RJ areas.

### 8.2.4 RS/RT Area

The RS area is an area that is 256 words long in the internal system RAM of the CPU. RS stores variable data of the system program and forms an interface between the microprogram and the system program. It occupies addresses "E F000H" to "E F0FFH."

#### NOTE

You should only write to system data words RS 60 to RS 63.

All other system data should only be read:

- Part of the RS system data area contains system variables and part of it contains information for the system programmer.
- Writing to the system data area can affect the functional capability of your programmable controller and connected programmers.

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 can use the entire RT area (RT 0 to RT 255), provided no standard function blocks are used. Otherwise this area may not be used.

Only an overall reset can clear the RS/RT areas.



## Memory Assignment and Memory Organization

### System Data of the RS Area

| RS                      | Name   | Address                       |
|-------------------------|--|-------------------------------|
| 0                       | Input byte "IB 0" image table (external process interrupts)            | EF000                         |
| 1                       | External process interrupts currently in the processing queue ("IB 0") | EF001                         |
| 2<br>•<br>•<br>•<br>4   | System program   |                               |
| 5                       | Current cycle time   | EF005                         |
| 6                       | System program   | EF006                         |
| 7                       | STOP mode ID (ISTACK)  | EF007                         |
| 8                       | Start/restart IDs (ISTACK)   | EF008                         |
| 9<br>•<br>•<br>•<br>15  | System program   |                               |
| 16                      | Error area: output bytes "0" to "15"                                   | EF010                         |
| 17<br>•<br>•<br>•<br>23 | Error area: output bytes "16" to "127"                                 | EF011<br>•<br>•<br>•<br>EF017 |
| 24<br>•<br>•<br>•<br>31 | Error area: input bytes "0" to "127"                                   | EF018<br>•<br>•<br>•<br>EF01F |
| 32<br>•<br>•<br>•<br>47 | Error area: interprocessor communication flag bytes "0" to "255"       | EF020<br>•<br>•<br>•<br>EF02F |
| 48<br>•<br>•<br>•<br>59 | System program   |                               |
| 60<br>•<br>•<br>•<br>63 | Available to user  | EF03C<br>•<br>•<br>•<br>EF03F |

| RS                        | Name  | Address |
|---------------------------|---|---------|
| 64<br>•<br>•<br>•<br>74   | System program                                  |         |
| 75                        | System stop code function number                | EF04B   |
| 76                        | System stop code parameter 1                    | EF04C   |
| 77                        | System stop code parameter 2                    | EF04D   |
| 78                        | System stop code parameter 3                    | EF04E   |
| 79<br>•<br>•<br>•<br>81   | System program                                  |         |
| 82                        | Interrupt condition code reset word (UALW) high | EF052   |
| 83                        | Interrupt condition code reset word (UALW) low  | EF053   |
| 84<br>•<br>•<br>•<br>95   | System program                                  |         |
| 96                        | Current time of day (seconds)                   | EF060   |
| 97                        | Current time of day (hours)                     | EF061   |
| 98                        | Current time of day (days)                      | EF062   |
| 99                        | Current time of day (year/month)                | EF063   |
| 100<br>•<br>•<br>•<br>252 | System program                                  |         |
| 253                       | free for distributed periphery                  | EF0FD   |
| 254<br><br>255            | System program                                  |         |

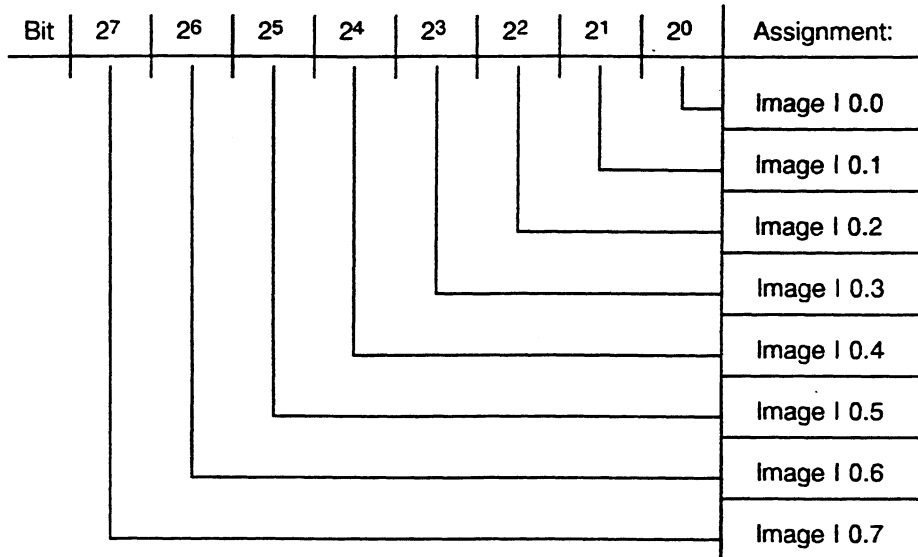
As a supplement to the listing above, the following pages provide the bit assignments of a few system data registers that you can evaluate via STEP 5 operations or with your programmer (see section 5.3 for information on the abbreviations).

## Memory Assignment and Memory Organization

### System Data: Register RS 0 - Input Byte "IB 0" Image Table (External Process Interrupts)

Address: E F000 (HIGH) is assigned to the system program

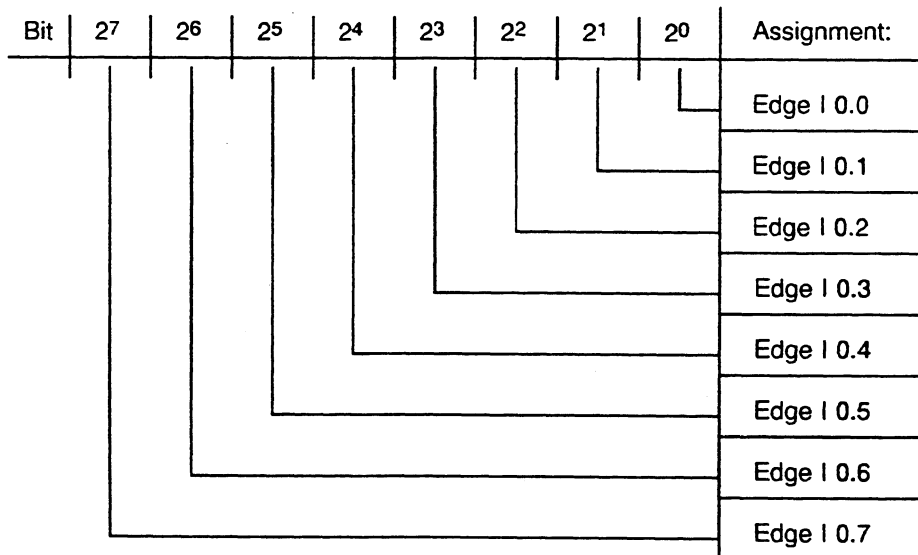
Address: E F000 (LOW)



### System Data: Register RS 1 - Condition Code of External Process Interrupts Currently in Processing Queue

Address: E F001 (HIGH) is '0'

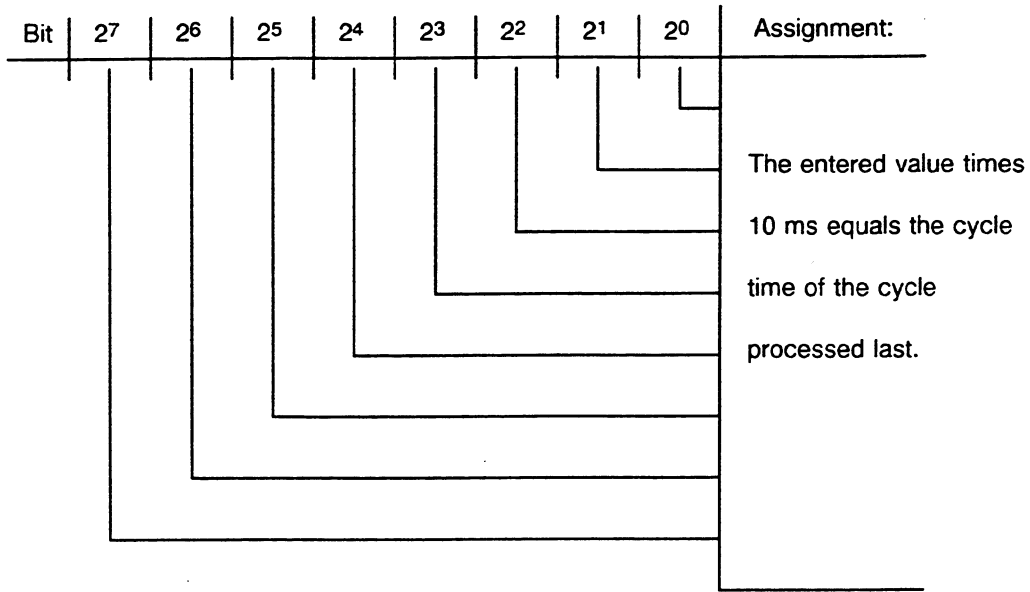
Address: E F001 (LOW)



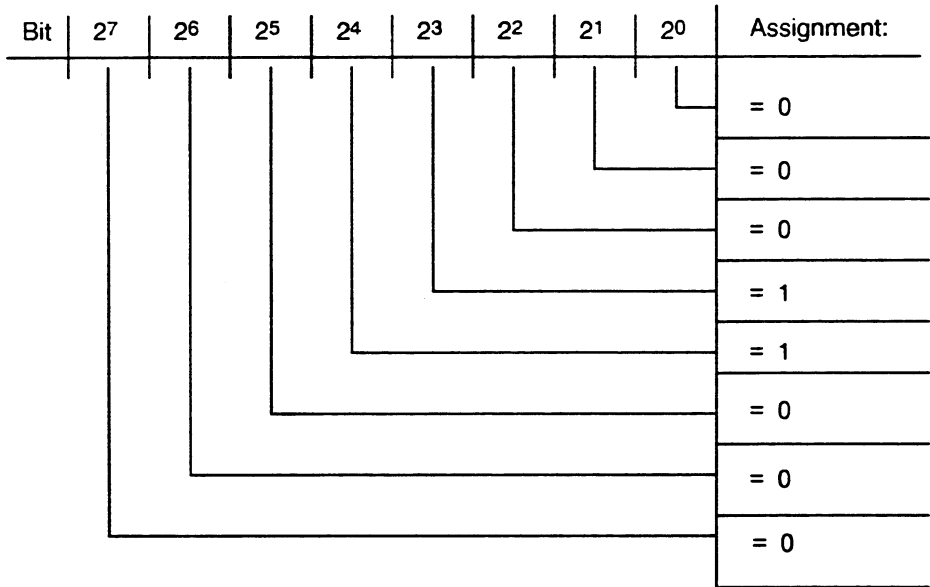
**System Data: Register RS 5 - Current Cycle Time**

Address: E F005 (HIGH) is '0'

Address: E F005 (LOW)



**Example:**

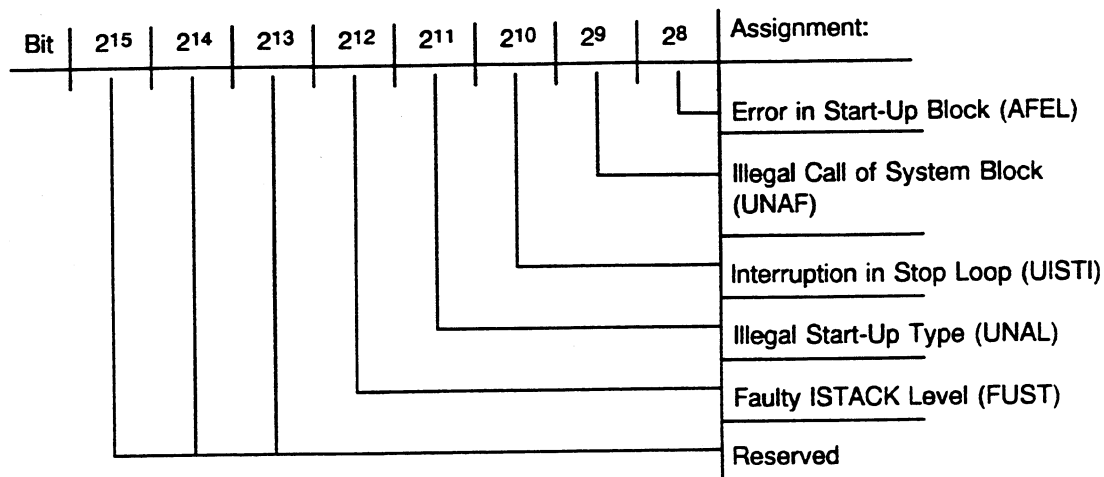


Bit 23 = 1, bit 24 = 1 and other bits = 0 means that the last cycle needed  $(8 + 16) * 10 \text{ ms} = 240 \text{ ms}$

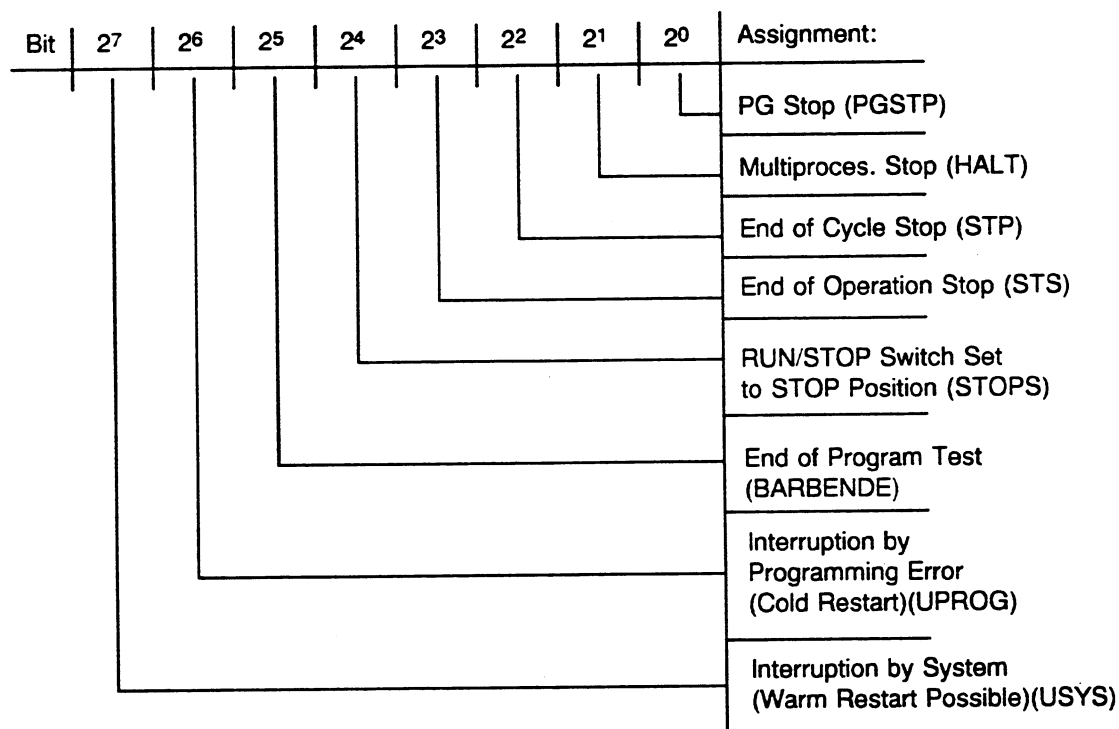
## Memory Assignment and Memory Organization

### System Data: Register RS 7 - Programmable Controller STOP Mode IDs (see ISTACK)

Address: E F007 (HIGH)

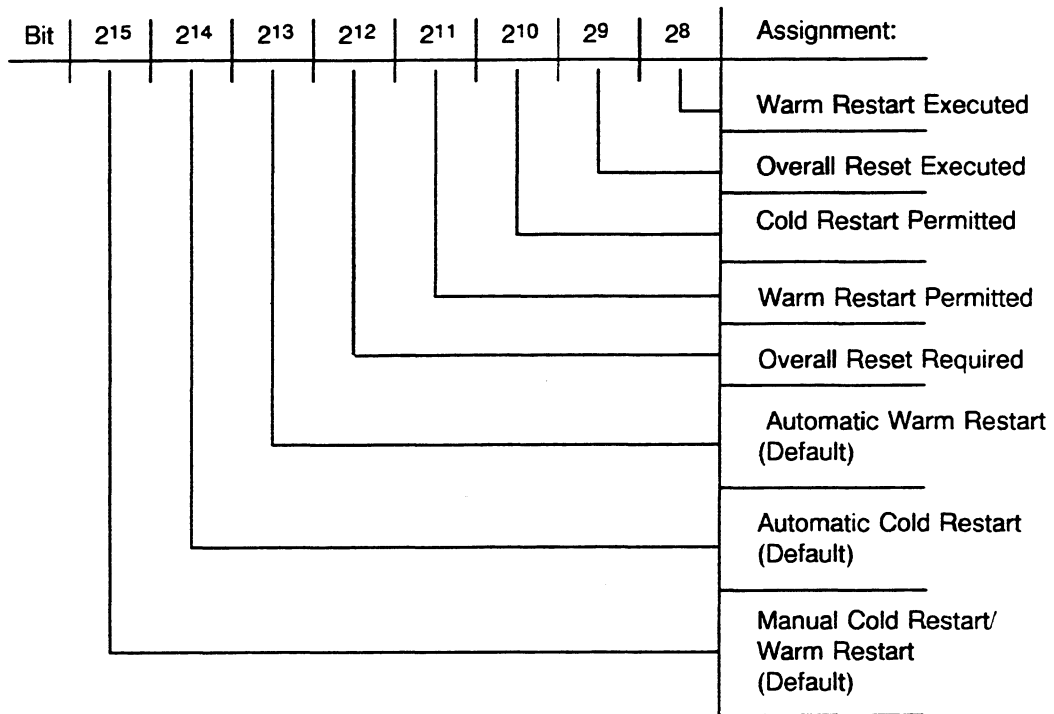


Address: E F007 (LOW)

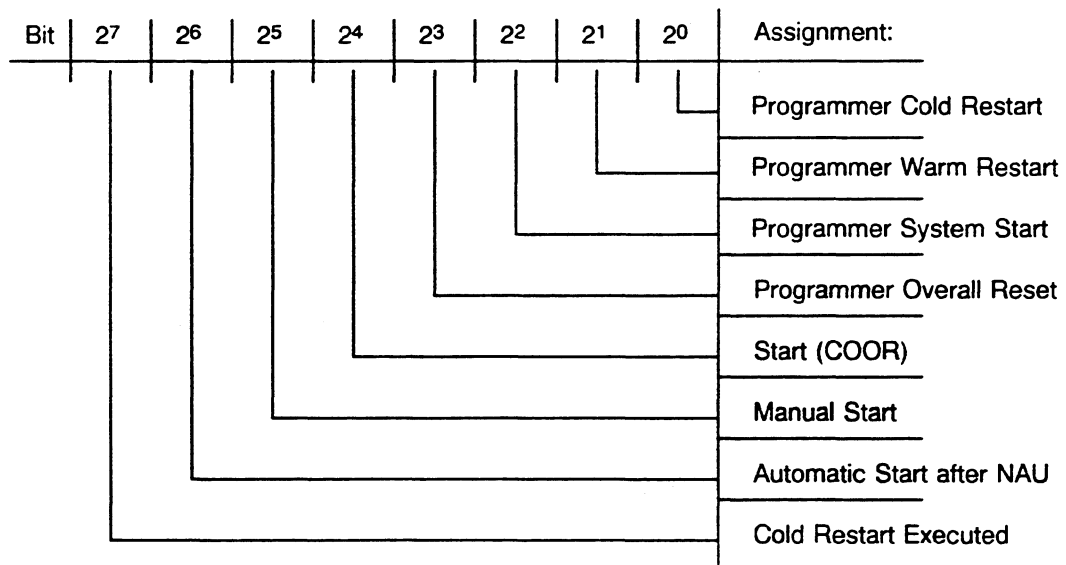


**System Data: Register RS 8 - Start and Restart IDs (see ISTACK)**

Address: E F008 (HIGH)

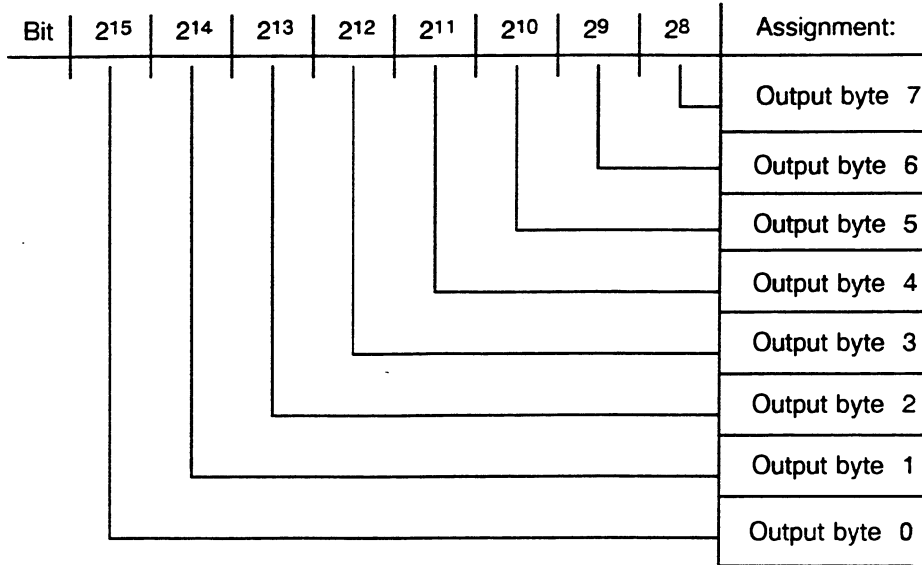


Address: E F008 (LOW)

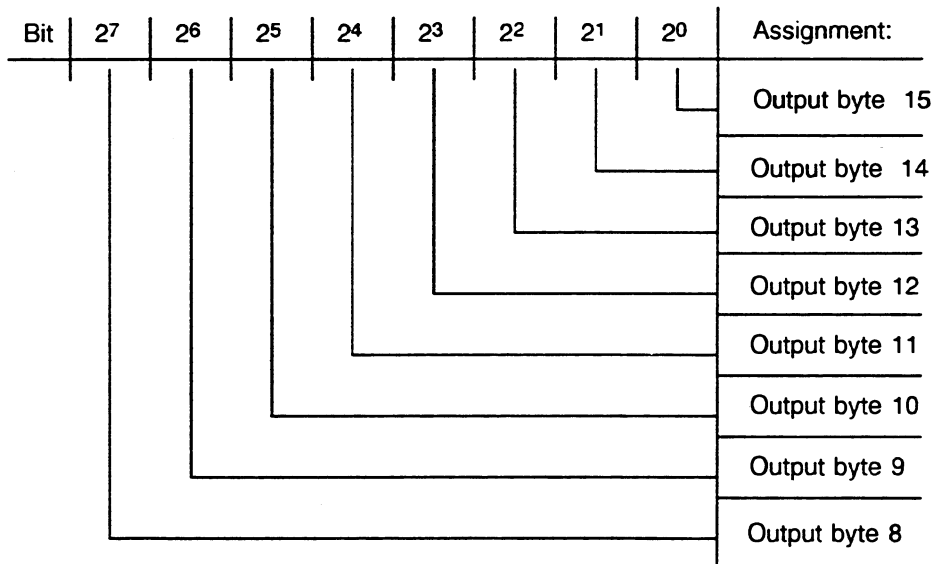


**System Data: Register RS 16 (and for RS 17 through RS 47)**

Address: E F010(HIGH)



Address: E F010 (LOW)



If errors appear during update of the process image input/output tables or interprocessor communication flags, the corresponding bits are set to 1. System data registers RS 17 to RS 47 have the same structure.

**Example**

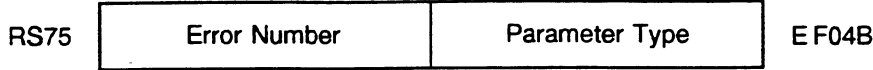
The content of system data register RS 16 is 8020 (hexadecimal) or 10000000 00100000 (binary).

The process image for output bytes 0 and 10 has not been updated correctly.

The entries in system data registers RS 75, RS 76, RS 77, and RS 78 refer to the errors that occurred last.

**System Data Register RS 75 - System Stop Code Function Number**

The function number consists of error number and parameter type.



The *high byte* contains the **error number** for the error that occurred. The error number assigns the error to one of the following three areas:

- 01H to 2FH: user error
- 30H to 3FH: error in DX 0 or DB 1
- 40H: system error

The *low byte* contains the **parameter type** that describes the structure of the succeeding parameter block. The parameter blocks are as follows:

- RS 76: parameter 1
- RS 77: parameter 2
- RS 78: parameter 3

Error numbers stored in system data register RS 75 for **user errors**.

| Error Number | Parameter Type | Error   |
|--------------|----------------|---|
| 01           | 01             | Block called is not loaded                                    |
| 02           | 01             | Addressing error  |
| 03           | 01             | Cycle time error  |
| 04           | 01             | Substitution error  |
| 05           | 02             | Timeout distributed peripherals                               |
| 06           | 03             | Timeout user memory   |
| 07           | 01             | Load/transfer error with data blocks and extended data blocks |
| 08           | 01             | Bracket counter overflow                                      |
| 09           | 04             | Opened data block does not exist.                             |
| 0A           | 05             | Error with internal time interrupts/other interrupts          |
| 0B           | 03             | Timeout page frame area                                       |
| 0C           | 03             | Timeout global communication area                             |
| 0D           | 07             | Timeout block transfer  |
| 0E           | 03             | Timeout IPC flags synchronization area                        |



## Memory Assignment and Memory Organization

| Error Number | Parameter Type | Error   |
|--------------|----------------|---|
| 0F           | 03             | Timeout "P"/"O" peripherals   |
| 10           | 03             | Timeout distributed interface module not plugged in   |
| 11           | 03             | Parity user memory  |
| 12           | 01             | Timeout process image transfer  |
| 13           | 01             | Timeout input byte "IB 0"   |
| 14           | 01             | BSTACK overflow   |
| 15           | 01             | STS operation   |
| 16           | 01             | RUN/STOP switch set to STOP position  |
| 17           | 01             | HALT signal from the coordinator  |
| 18           | 01             | Timeout process image transfer  |
| 19           | 01             | Load/transfer error with L BY/T BY (process image update)   |
| 1A           | 01             | Load/transfer error during addressing via the BR register   |
| 1B           | 01             | I/O not ready   |
| 1C           | 06             | Timeout/parity error during initialization  |
| 1D           | 08             | Automatic warm restart not possible; cold restart required  |
| 1E           | 00             | Illegal start-up type   |
| 1F           | 01             | Load/transfer error during block move operation (incorrect memory area boundaries with TNW, TXB, and TXW) |
| 20           | 09             | Illegal length for G DB/GX DX   |
| 21           | 09             | DB/DX already exists for G DB/GX DX   |
| 22           | 09             | Memory space insufficient for G DB/GX DX  |
| 23           | 05             | Masked interrupt is coming through  |
| 24           | 00             | Block function (compression, transfer, input) in STOP; cold restart required                              |
| 25           | 00             | Battery failure; start-up not possible  |

Error numbers in RS 75 for errors in DX 0.

| Error Number | Parameter Type | Error   |
|--------------|----------------|---|
| 30           | 00             | No DX0 in multiprocessing   |
| 31           | 00             | No block with block ID 01 (155U controller mode) in multiprocessing |
| 32           | 00             | 150U controller mode is set for multiprocessing                     |
| 33           | 00             | Interrupts and PLC mode not compatible                              |
| 34           | 06             | Invalid DX 0 header   |
| 35           | 07             | Error in DX 0 block ID  |
| 36           | 07             | Error in DX 0 parameter   |

Error numbers stored in RS 74 for errors in DB 1.

| Error Number | Parameter Type | Error  |
|--------------|----------------|--|
| 38           | 06             | No DB 1 for multiprocessing                  |
| 39           | 06             | Invalid DB 1 header                          |
| 3A           | 07             | DB 1 ID set more than once                   |
| 3B           | 07             | DB 1 byte offset without ID                  |
| 3C           | 07             | Peripheral entered in DB 1 is not plugged in |
| 3D           | 07             | Offset too big (parameter error)             |
| 3E           | 07             | Too many offsets                             |

The *low byte* of RS 75 contains the **parameter type** for each case. The parameter type defines the structure of the parameter block that follows (RS 76 to RS 78):

| Parameter Type | Structure of the Parameter Block   |
|----------------|--|
| 00H            | No parameter; parameter 1, 2, 3 = 0  |
| 01H            | Parameter 1: block type/block number (IDs from block header)<br>Parameter 2: operation that caused an interruption   |
| 02H            | Parameter 1: interface module number (IM 302) (distributed periphery)<br>Parameter 2: number of the defective interface<br>Parameter 3: incorrect byte offset  |
| 03H            | Parameter 1: memory module/submodule number (FFFFH, if no module and no submodule can be assigned)<br>Parameter 2: error address high<br>Parameter 3: error address low  |
| 04H            | Parameter 1: block type/block number (IDs from block header)<br>Parameter 2: operation that caused the interruption<br>Parameter 3: number of the opened DB/DX   |
| 05H            | Error during internal time interrupt processing or HW signal interrupt processing<br>Parameter 1: The following bits are set depending on the error:<br>Bit 0 ≐ period 1<br>Bit 1 ≐ period 2<br>: : :<br>Bit 7 ≐ period 8<br>Bit 8 ≐ period 9<br>Bit 9 = 1: clock masked (ignored) for too long<br>= 0: queue overflow<br>Bit 10, 11: reserved<br>Bit 12 ≐ interrupt G<br>Bit 13 ≐ interrupt F<br>Bit 14 ≐ interrupt E<br>Bit 15 ≐ interrupt X |
| 06H            | Error in DX 0 or DB 1 header<br>Parameter 1: data word setpoint<br>Parameter 2: actual data word   |
| 07H            | Error in DX 0 or DB 1 interpretation and for QVZ during update of process image<br>Parameter 1: block ID or code word (see DX 0/DB 1)<br>Parameter 2: incorrect parameter 1 ('FFFFH': parameter irrelevant)<br>Parameter 3: incorrect parameter 2 ('FFFFH': parameter irrelevant)  |

| Parameter Type | Structure of the Parameter Block  |
|----------------|---|
| 08H            | Error during 80186 initialization<br>Parameter 1: Bit 0 ≐ QVZ in initialization<br>Bit 1 ≐ PARE in initialization<br>Bit 2 ≐ reserved<br>Bit 3 ≐ RMX error<br>Bit 4 ≐ incorrect block ID<br>Bit 5 ≐ incorrect block delimiter<br>Bit 6 ≐ memory modules different<br>Bit 7 ≐ memory submodules different<br>Bit 8 ≐ DB 0 different<br>Bit 9 to 15: reserved<br>Parameter 2: 0<br>Parameter 3: 0 |
| 09H            | Error for the G DB or GX DX operation<br>Parameter 1: block type (machine code)<br>Parameter 2: block number<br>Parameter 3: data block length  |
| 10H            | Internal system error number  |

The following example evaluates system data registers RS 75 to RS 78:

**Example:**

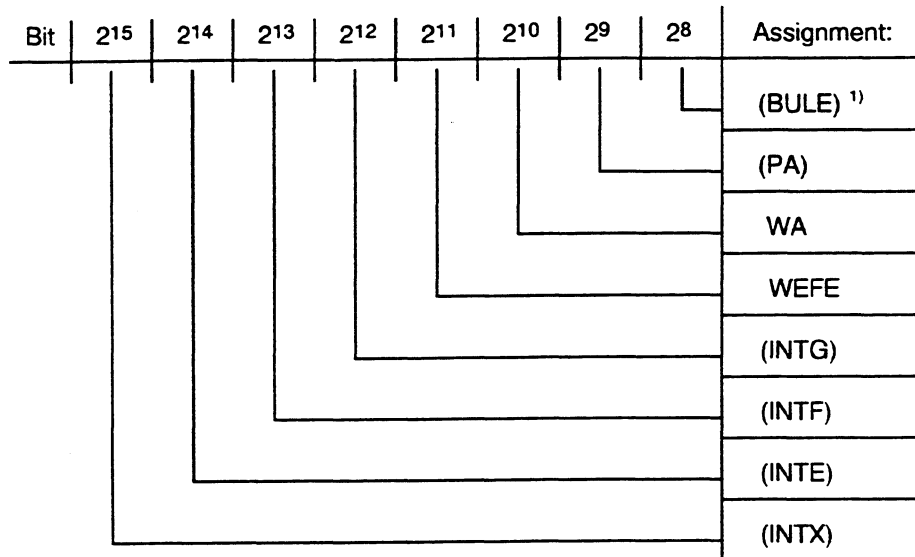
|       |        |  |        |        |
|-------|--------|--|--------|--------|
| RS 75 | 2    1 |  | 0    9 | E F04B |
| RS 76 | 7    8 |  | 0    4 | E F04C |
| RS 77 | 0    0 |  | 6    4 | E F04D |
| RS 78 | 0    0 |  | 7    8 | E F04E |

- RS 75, error number "21":            The error occurred in the STEP 5 user program.
- RS 75, parameter type "09":        Error in the G DB/GX DX operation; DB/DX already exists; RS 76, RS 77, and RS 78 must be evaluated.
- RS 76, parameter 1:                Contains the block type (machine code, first word); 7805 = DX data block.
- RS 77, parameter 2:                Contains the block number; 0064 = DX 100.
- RS 78, parameter 3:                Contains the number of data words stored; 0078 = 120 data words.

## Memory Assignment and Memory Organization

**System Data: Register RS 82 - Interrupt Condition Code Reset Word (UALW High Word, High Byte)**

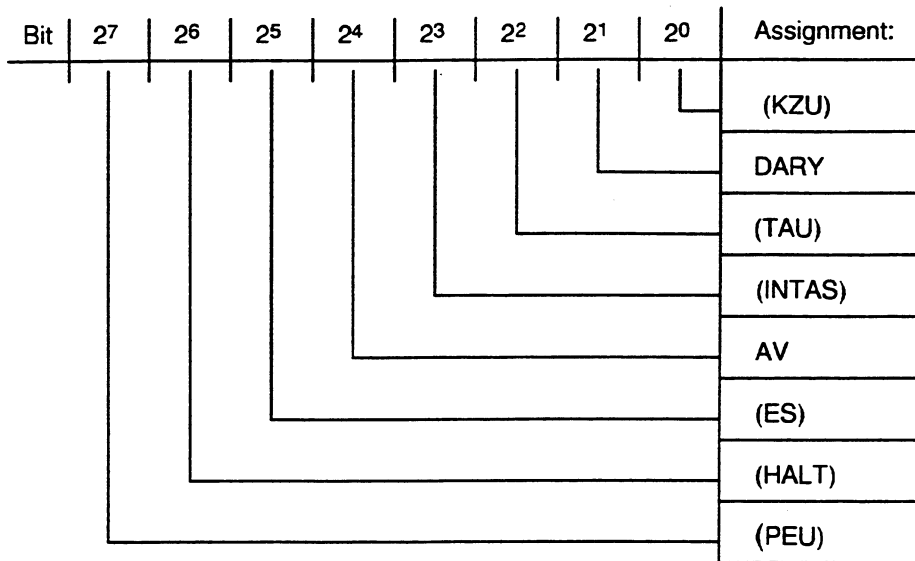
Address: E F052 (HIGH)



**1) Note: Interrupts in brackets cannot be reset.**

- **BULE:**  
bus-lock error
- **PA (external process interrupt):**  
An interrupt signal was detected at an operation boundary, triggered by a signal status change in input byte "IB 0."
- **WA (internal time interrupt):**  
An internal time interrupt was detected at an operation boundary, triggered by the internal time interrupt basic clock rate.
- **WEFE (collision of time interrupts):**  
The internal clock was masked (ignored) for too long.
- **INTG, INTF, INTE, INTX (HW signal interrupts).**  
An interrupt signal was detected at an operation boundary, triggered by the following system interrupts:  
A or B at INTX (slot dependent)  
E at INTE  
F at INTF  
G at INTG

Address: E F052 (LOW)

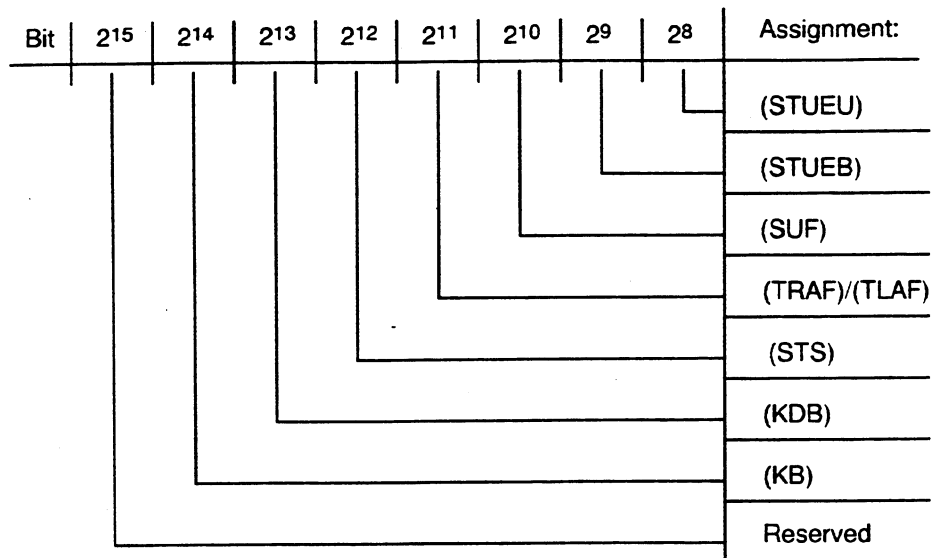


- **KZU:**  
The bracket stack has overflowed (more than eight brackets).
- **DARY:**  
Access was made to a faulty memory area.
- **TAU:**  
The 80186 microprocessor clock pulse failed.
- **INTAS:**  
There was an interrupt from the 80186 microprocessor.
- **AV :**  
The address comparator is active.
- **ES:**  
The program is being processed in "SINGLE-STEP" mode (with the STATUS or PROGRAM TEST programmer function).
- **HALT:**  
A STOP mode indication is coming from the coordinator in multiprocessing.
- **PEU:**  
There is a power failure at an expansion unit.

## Memory Assignment and Memory Organization

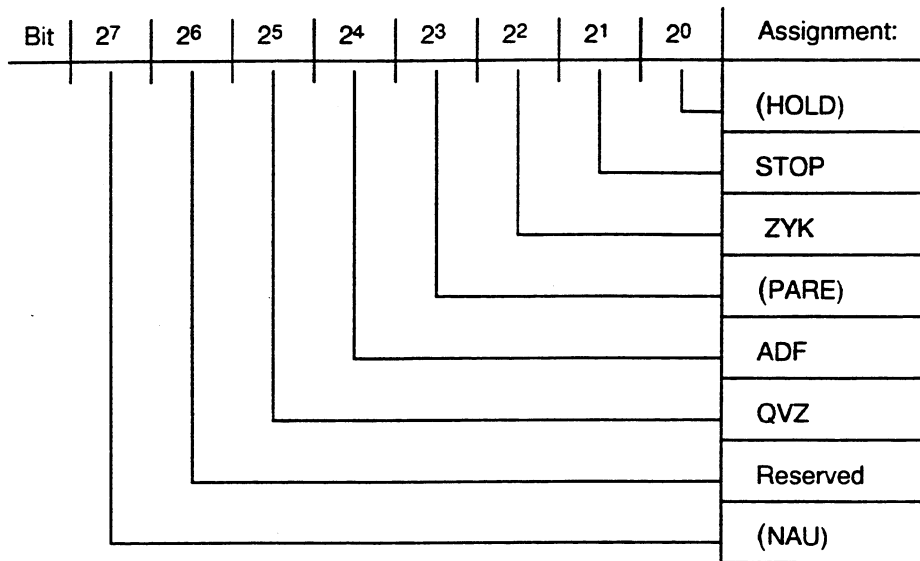
**System Data: Register RS 83 - Interrupt Condition Code Reset Word (UALW Low Word, High Byte)**

Address: E F053 (HIGH)



- **STUEU:**  
ISTACK overflow
- **STUEB:**  
BSTACK overflow
- **SUF:**  
substitution error
- **TRAF/TLAF:**  
transfer or load error during access to data blocks
- **STS:**  
STOP mode indication (exact statement stop) by STEP 5 operation in the user program
- **KDB:**  
call of a DB/DX that is not loaded
- **KB:**  
call of a programmable logic block that is not loaded

Address: E F053 (LOW)



- **HOLD:**  
DMA request from the 80186 microprocessor
- **STOP:**  
RUN/STOP switch set to the STOP position
- **ZYK:**  
cycle time exceeded
- **PARE:**  
parity error
- **ADF:**  
addressing error
- **QVZ:**  
timeout
- **NAU:**  
power failure



## Real-Time Clock in the S5-155U: RS 96 to RS 99

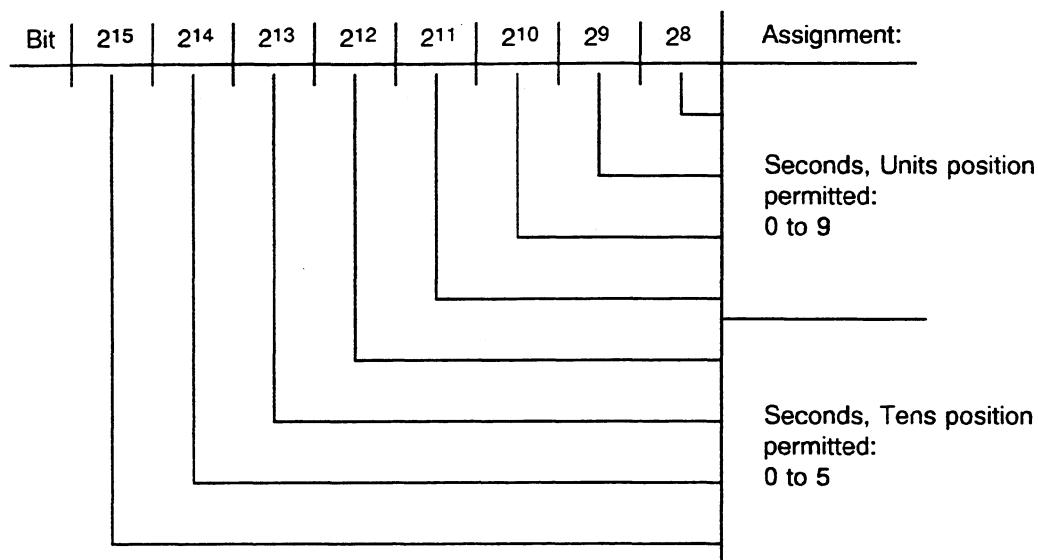
The current date and time of day are kept in system data registers RS 96 to RS 99 as follows:

- RS 96: seconds (current time of day)
- RS 97: hours and minutes (current time of day)
- RS 98: date and day of the week (current time of day)
- RS 99: year/month (current time of day)

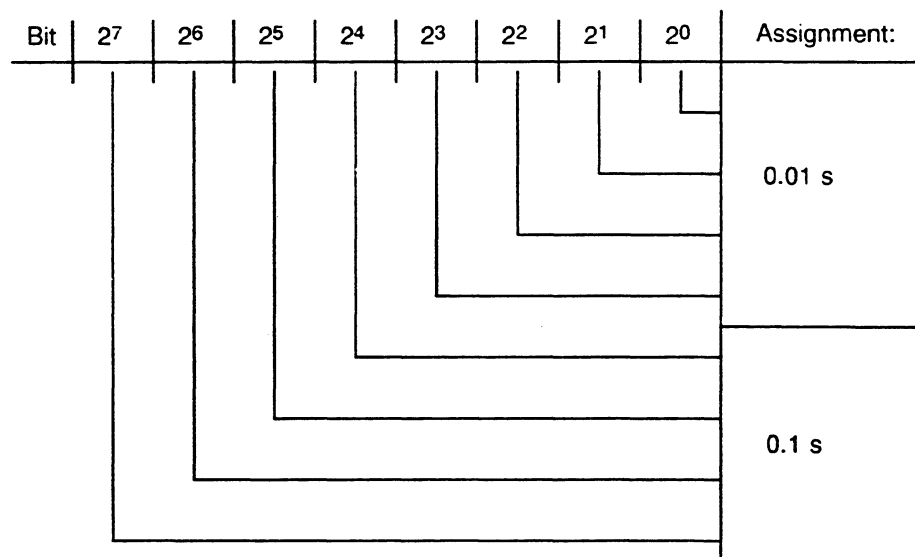
The clock is updated in a 10 ms pulse if you enable time of day register processing. Otherwise, the time is updated in a second's pulse only.

### RS 96 - Current time of day in seconds

E F060 (HIGH)

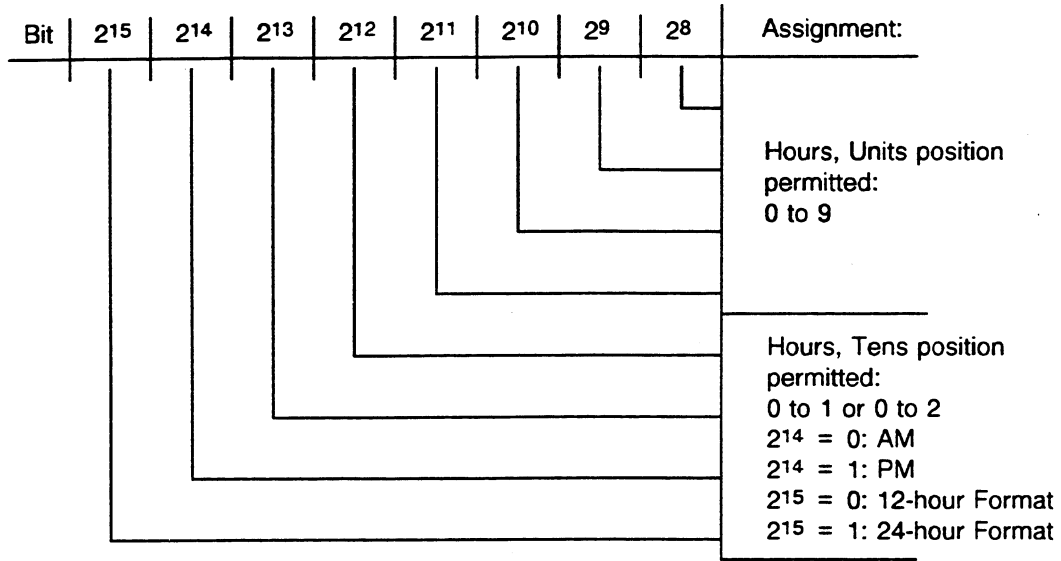


E F060 (LOW)

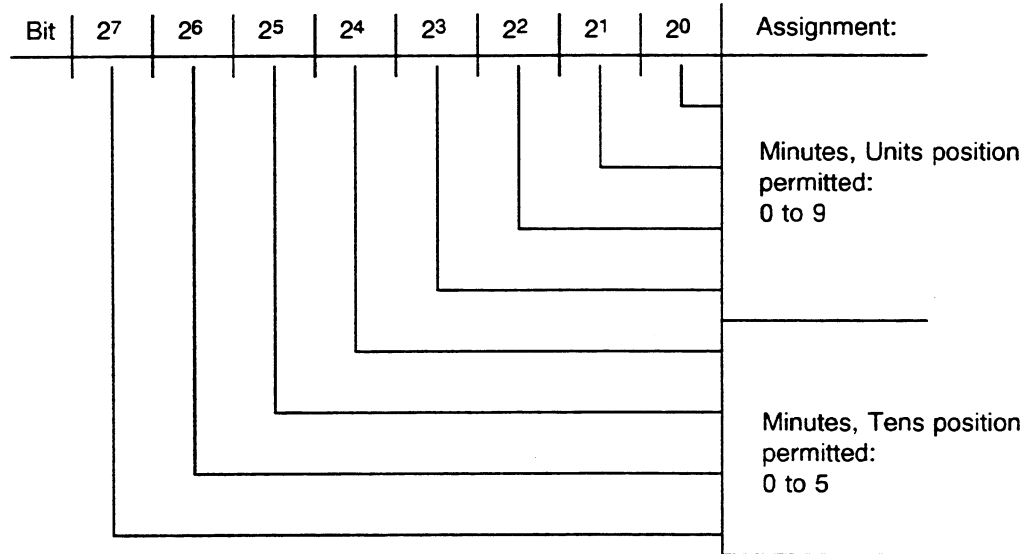


RS 97 - Current time of day in hours and minutes.

E F061 (HIGH)



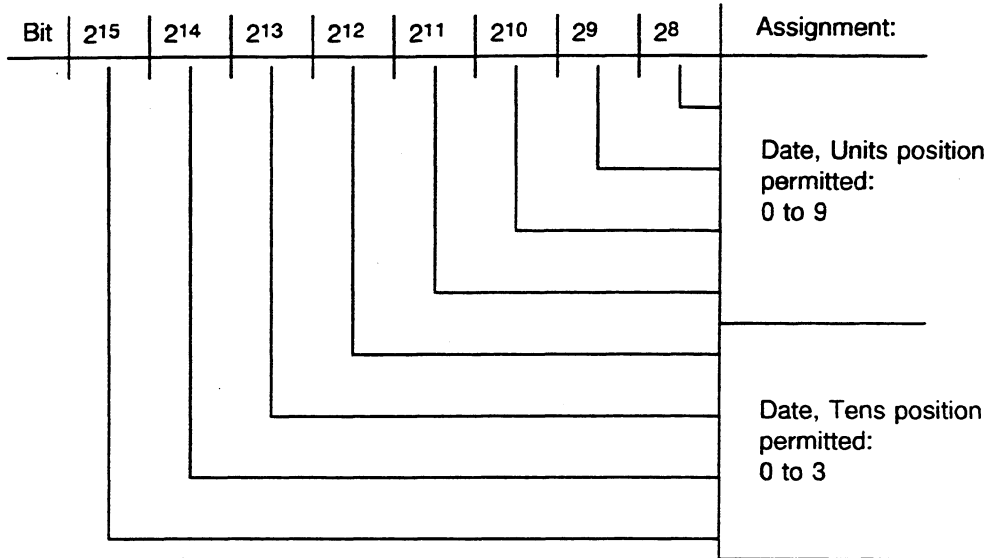
E F061 (LOW)



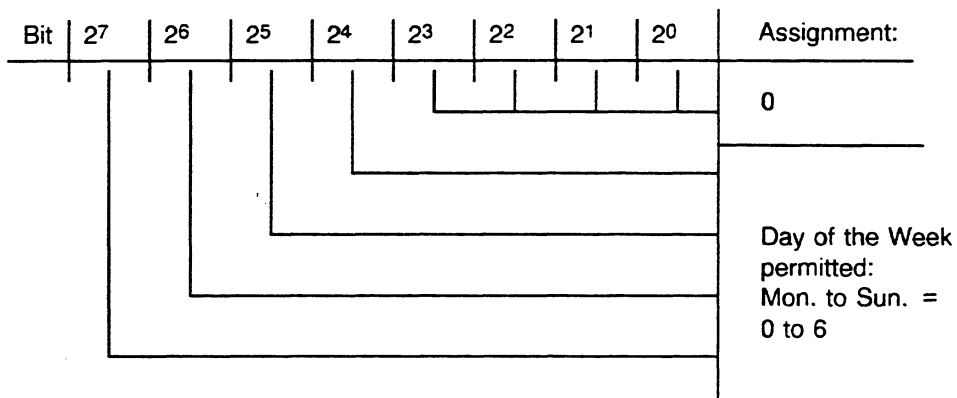
# Memory Assignment and Memory Organization

## RS 98 - Current date and day of the week

E F062 (HIGH)

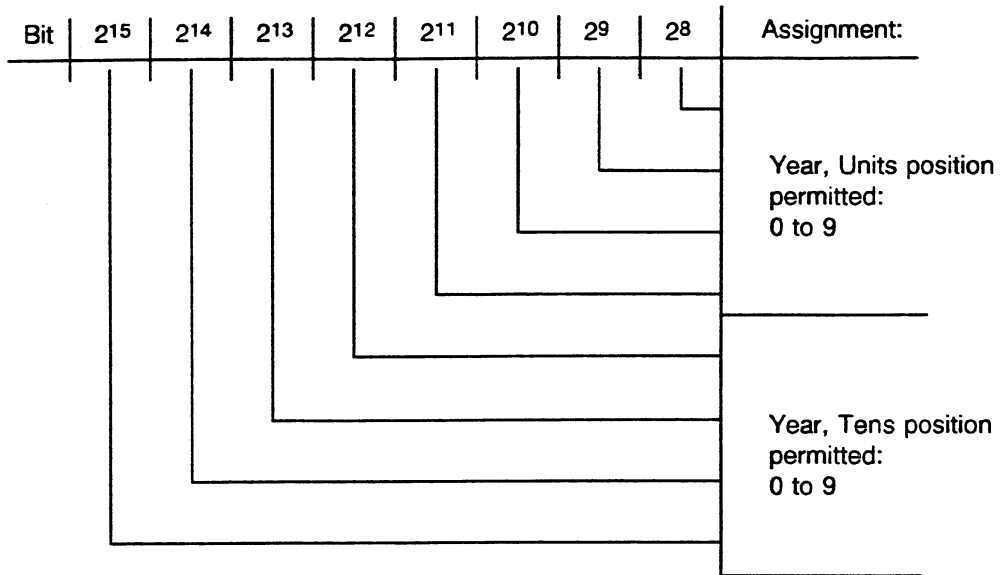


E F062 (LOW)

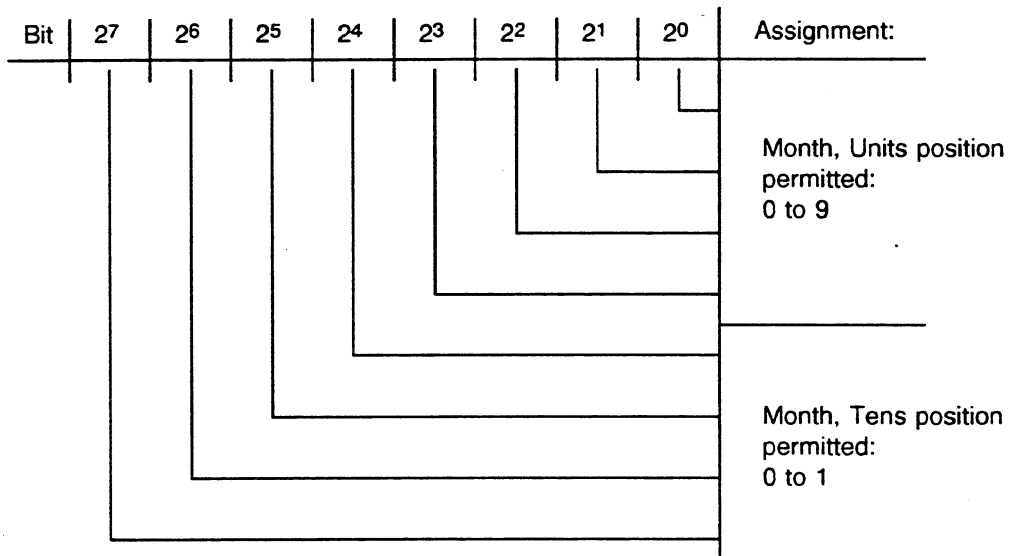


RS 99 - Current year and month

E F063 (HIGH)



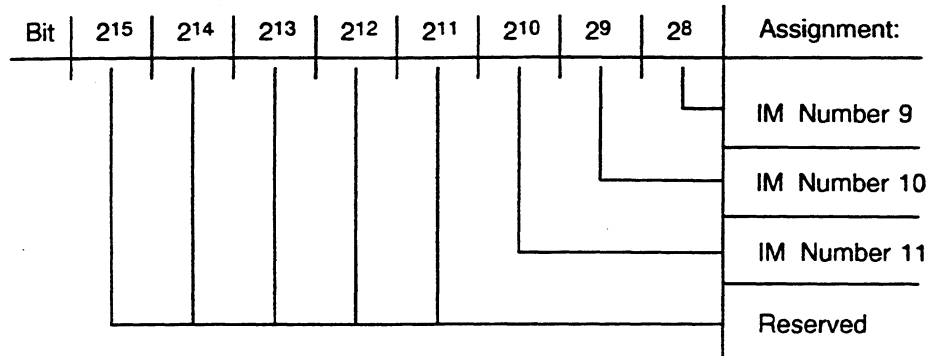
E F063 (LOW)



## Memory Assignment and Memory Organization

### System Data: Register RS 253 - Directory of Plugged-In Interface Modules

Address: E F0FD (HIGH)



Address: E F0FD (LOW)



**8.2.5 Addressable System Data Area**

The system program uses the memory area from "E 8200H" to "E DEF0H" as an addressable system data area.

**Programmable Controller Identification Block**

|         |     |     |     |     |         |
|---------|-----|-----|-----|-----|---------|
| Word    |     |     |     |     |         |
| 0 - 1   | 'S' | '5' | '1' | '5' | E 8200H |
| 2 - 3   | '5' | 'U' | 'V' | '1' |         |
| 4 - 5   | '.' | '0' |     | 0   |         |
| 6 - 7   |     | 0   |     | 0   |         |
|         |     |     |     |     |         |
| 10 - 11 |     | 0   |     | 0   | E 820BH |

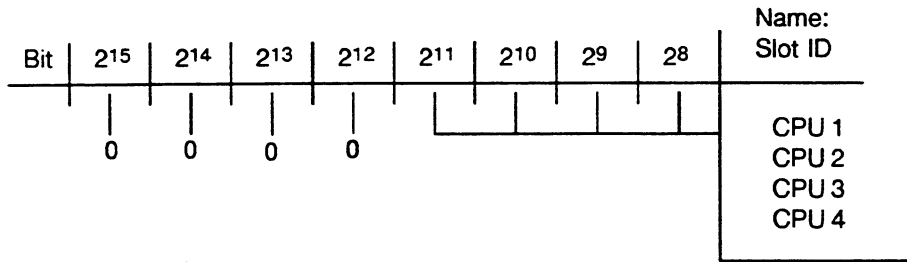
System parameters in the memory area beginning with the address "E 8210H."

|      |                            |   |         |
|------|----------------------------|---|---------|
| Word |                            |   |         |
| 0    | Interface Module Input     |   | E 8210H |
| 1    | Interface Module Output    |   |         |
| 2    | Process Image Input Table  |   |         |
| 3    | Process Image Output Table |   |         |
| 4    | Flags                      |   |         |
| 5    | Timers                     |   |         |
| 6    | Counters                   |   |         |
| 7    | System Data                |   |         |
| 8    | Status ID                  | Program. Controller<br>Software Version |         |
| 9    | User Memory End Address    |   |         |
| 10   | System Program Memory      |   |         |
| 11   | Length of the DB List      |   |         |
| 12   | Length of the SB List      |   |         |
| 13   | Length of the PB List      |   |         |

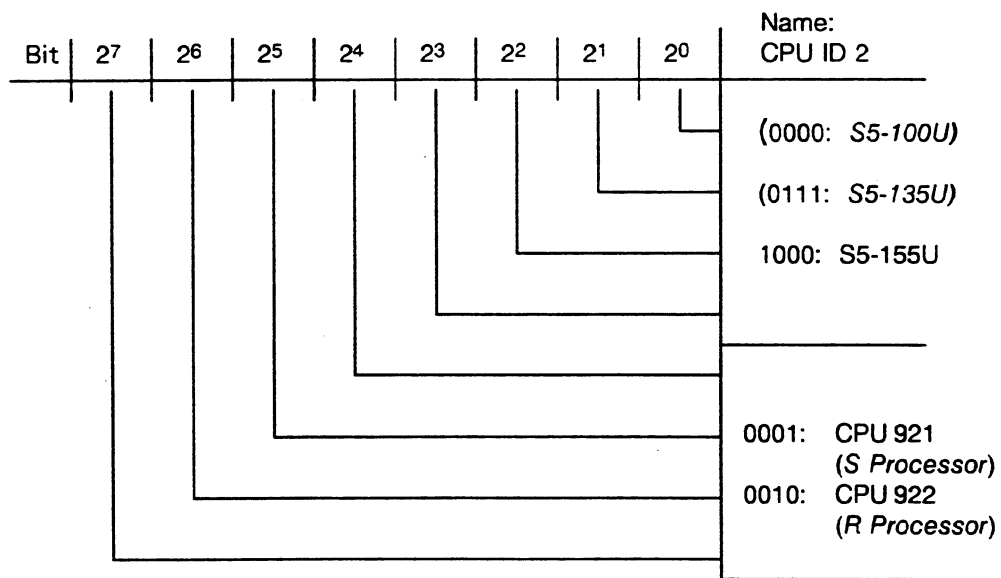
|    |                                      |                                       |         |
|----|--------------------------------------|---------------------------------------|---------|
| 14 | Length of the FB List                |                                       | E 8225H |
| 15 | Length of the OB List                |                                       |         |
| 16 | Length of the FX List                |                                       |         |
| 17 | Length of the DX List                |                                       |         |
| 18 | Length of the DB Address List (DB 0) |                                       |         |
| 19 | Slot ID 1)                           | CPU ID 2                              |         |
| 20 | Block Header Length                  |                                       |         |
| 21 | CPU ID 1)                            | Programmer Interface Software Version |         |

1) You can also use the SYSTEM PARAMETER programmer 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).

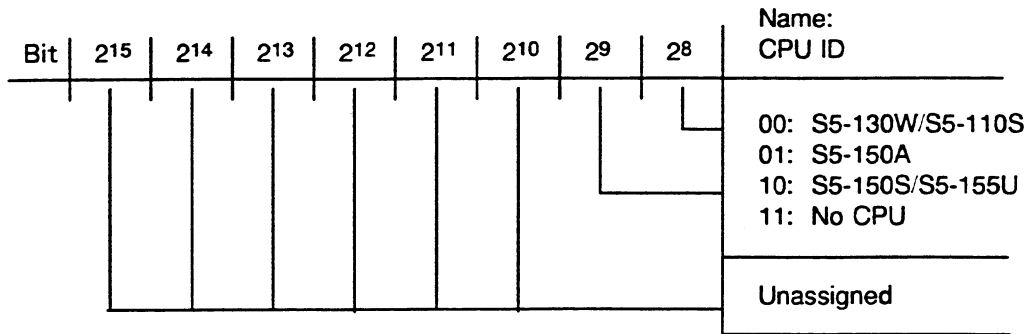
**WORD 19 (HIGH)**



**WORD 19 (LOW)**



**WORD 21 (HIGH)**



**WORD 21 (LOW)**

Low word 21 contains the programmer interface software (PGAS) version written in hexadecimal code (e.g., 13 in hexadecimal code indicates programmer interface software with revision level 1.3).



## Chapter 9

# Memory Access Using Absolute Addresses

The STEP 5 programming language contains system operations with which you can access the entire memory area.

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.

### NOTE

**If the operations described in this chapter are not used properly, STEP 5 blocks and system data can be overwritten. This can result in undesirable operating statuses. Therefore, only experienced programmers should use system 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 image tables.

#### **Global Memory**

Global memory is a common memory area shared by all CPUs in multiprocessing. You address it via the S5 bus.

#### **Memory Organization**

Memory areas are organized in bytes or in words as follows:

**Bytes:** Each register addresses a byte.

**Words:** Each register addresses a word (one word is approximately equal to two bytes).

Organization of the local memory is prescribed (see Chapter 8). Organization of the global memory depends on the type of modules that are plugged into the programmable controller.

## Memory Access Using Absolute Addresses

The local memory is internal and is available in each CPU (acc. to the number of CPUs plugged)

The global memory is external and is available via the S5 bus. It exists as a common memory area shared by all CPUs in one PLC.

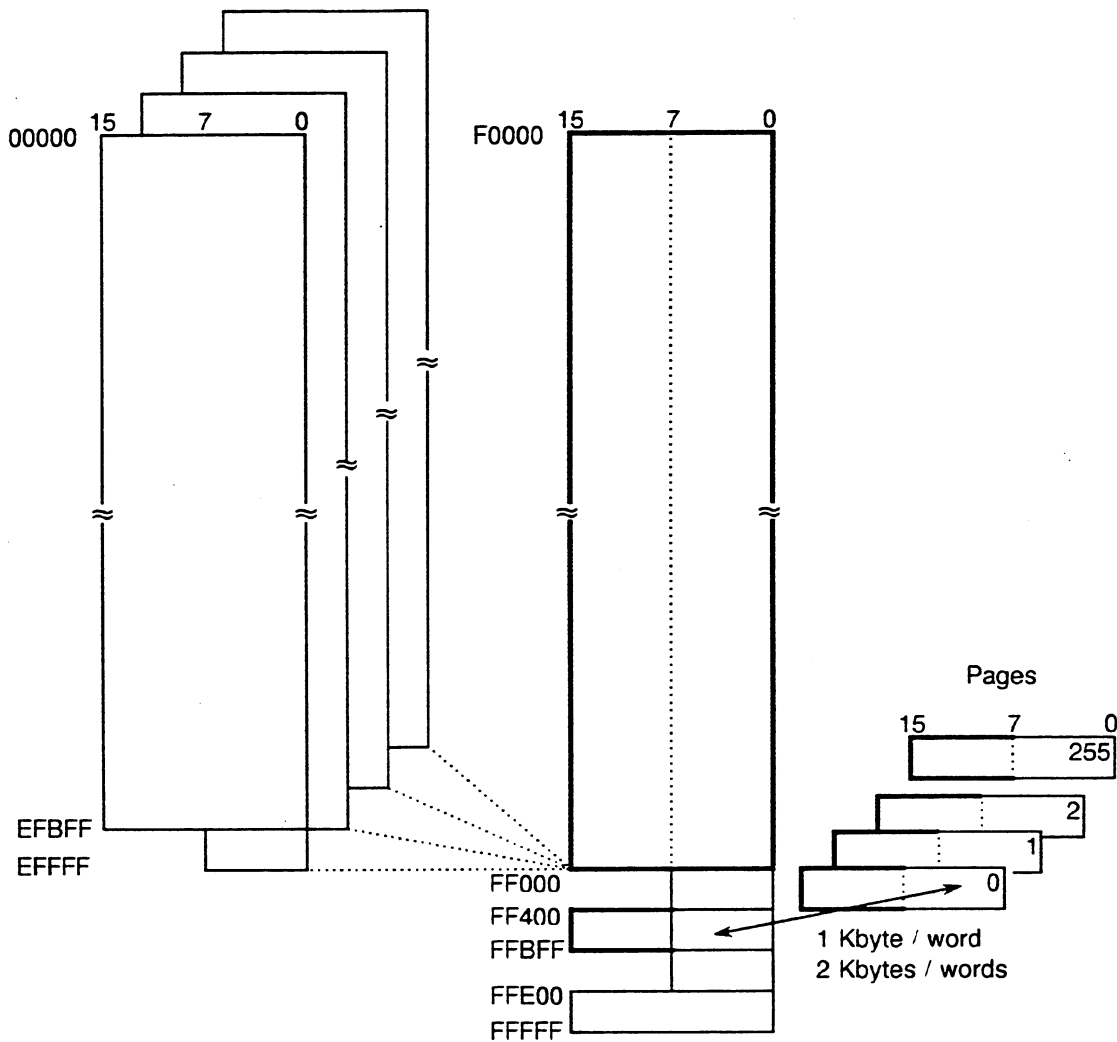


Fig. 9.1 Local and Global Memory

Using *absolute addresses*, you can access the following local or global memory areas with the operations indicated:

- Local area (00000 to EFFFF) and global area (F0000 to FFFFF) - LIR, TIR, LDI, TDI, TNW, TXB, TXW
- Portion of the local area organized in words (00000 to EFBFF) - LRW, TRW, LRD, TRD

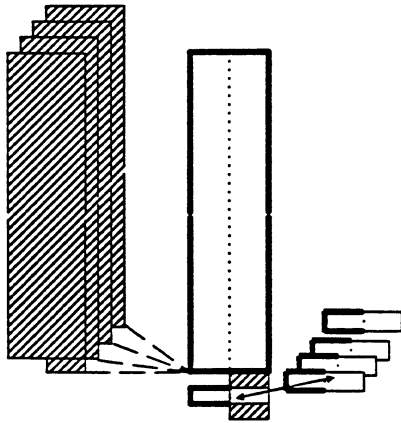
- c) Portion of the global area organized in bytes (F0000 to FFFFF) - LY GB, LY GW, LY GD, TY GB, TY GW, TY GD, TSG
- d) Portion of the global area organized in words (F0000 to FFFFF) - LW GW, LW GD, TW GW, TW GD, TSG
- e) Portion of the global area organized in bytes (FF400 to FFBBF, dual-port RAM area) - LY CB, LY CW, LY CD, TY CB, TY CW, TY CD, TSC
- f) Portion of the global area organized in words (FF400 to FFBBF, dual-port RAM area) - LW CW, LW CD, TW CW, TW CD, TSC

# Memory Access Using Absolute Addresses

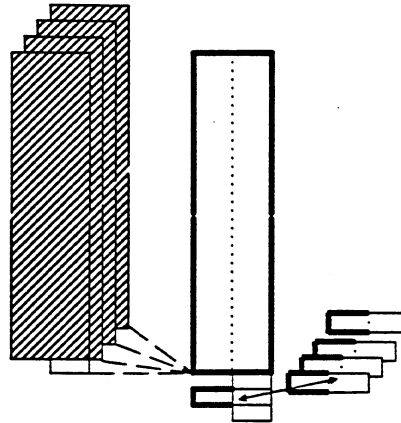
Access to local or global memory areas via absolute addressing  
(see also Fig. 9.1)

□ access not possible

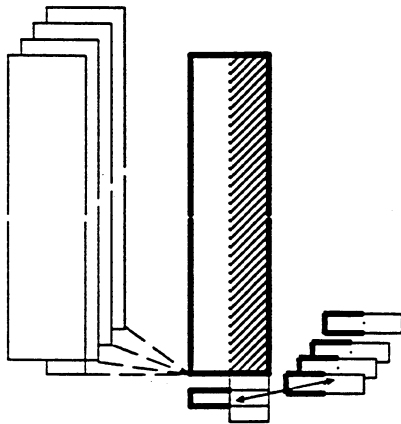
▨ access possible



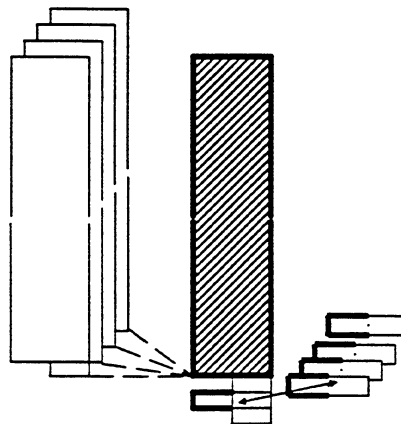
a) LIR, TIR, TNB, TNW



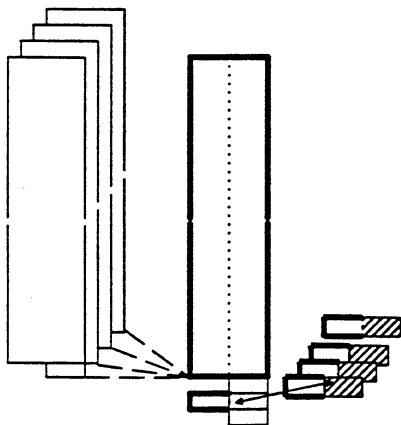
b) LRW, TRW, LRD, TRD



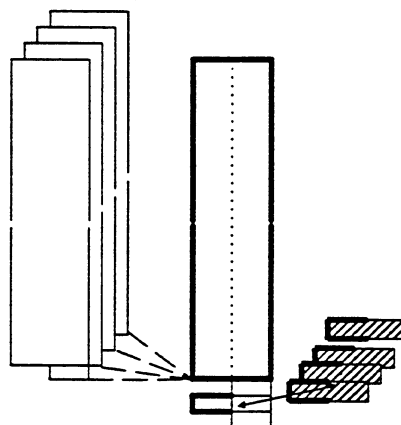
c) LY GB, LY GW, LY GD  
TY GB, TY GW, TY GD, (TSG)



d) LW GW, LW GD  
TW GW, TW GD, (TSG)



e) LY CB, LY CW, LY CD  
TY CB, TY CW, TY CD (TSC)



f) LW CW, LW CD,  
TW CW, TW CD, (TSC)

### 9.1 Access to Registers Using ACCU 1

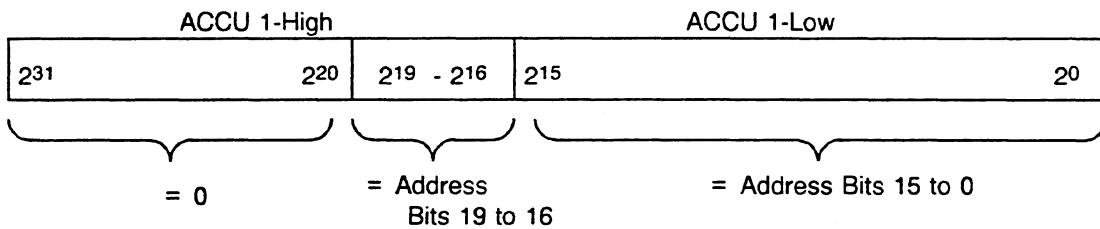
Registers are internal memory cells that the CPU needs to process a STEP 5 program. Registers have the following widths: 16 bits, 32 bits. You can use the following system operations to access the register contents or to load/transfer the memory register contents: load the contents of a register indirectly (LIR), transfer the contents of a register indirectly (TIR), load the contents of a double-word register indirectly (LDI), transfer the contents of a double-word register indirectly (TDI).

| Operation | Parameter     | Explanation   |
|-----------|---------------|---|
| LIR       | 0 to 15       | Load the contents of the memory register whose address is located in ACCU 1-High/Low (20 bits) into the 16-bit register addressed with the parameter value.                                 |
| TIR       | 0 to 15       | Transfer the contents of a 16-bit register to the memory register whose address is located in ACCU 1-High/Low (20 bits).  |
| LDI       | Register name | Load the contents of memory register n whose address is located in ACCU 1-High/Low (20 bits) and the contents of the next memory register (memory register n + 1) into the 32-bit register. |
| TDI       | Register name | Transfer the contents of a 32-bit register to the memory register n whose address is located in ACCU 1-High/Low (20 bits) and to the next register (n + 1).                                 |

**NOTE**

**Do not use the TIR and TDI operations to access EPROM submodule addresses. Doing so overwrites the parity memory bits.**

Bit assignment in ACCU 1-High/Low of an absolute address of a memory register.



Never use the LIR, TIR, LDI, and TDI operations to access STEP 5 programmable logic blocks (OBs, FBs, PBs, and SBs). Doing so can result in undesirable operating statuses. Use them for accessing data blocks or one of the other operand areas.

The next pages provide an explanation of the individual registers and some programming examples for the LIR, TIR, LDI, and TDI operations.

## Memory Access Using Absolute Addresses

---

### LIR and TIR: Loading to or Transferring from a 16-Bit Memory Area Indirectly

The following table shows the register assignment for the operations LIR and TIR for the CPU 946/947.

| Register Number (Parameter) | Register Assignment (Each 16 Bits Wide)                        |
|-----------------------------|--|
| Register 0:                 | ACCU 1-High (left word of ACCU 1, bits 16 to 31) <sup>1)</sup> |
| Register 1:                 | ACCU 1-Low (right word of ACCU 1, bits 0 to 15) <sup>1)</sup>  |
| Register 2:                 | ACCU 2-High  |
| Register 3:                 | ACCU 2-Low   |
| Register 4:                 | -  |
| Register 5:                 | Block stack pointer (offset)                                   |
| Register 6:                 | Data block start address (DBA) register                        |
| Register 7:                 | -  |
| Register 8:                 | Data block length (DBL) register                               |
| Register 9:                 | ACCU 3-High  |
| Register 10:                | ACCU 3-Low   |
| Register 11:                | ACCU 4-High  |
| Register 12:                | ACCU 4-Low   |
| Register 13:                | Auxiliary register 1 <sup>2)</sup>                             |
| Register 14:                | Auxiliary register 2 <sup>2)</sup>                             |
| Register 15:                | -  |

1) Loading the contents of an addressed memory register into register 0 overwrites the address stored in ACCU 1-High/Low.

2) The microprogram uses the auxiliary registers. You should not use them, since their contents are not carried over to the next operation.

### LIR and TIR: Loading to or Transferring from an 8-Bit Memory Area Indirectly

Note the following if you use the LIR and TIR operations to access memory areas that are only 8 bits wide (i.e., for memory addresses E FC00H and higher: flags, PII, PIQ):

- The TIR operation transfers only the low byte of the register. The high byte of the register is lost.
- The LIR operation writes nondefined values to the high byte of the register.

**Registers 0 to 3 and 9 to 12: ACCU 1, ACCU 2, ACCU 3, and ACCU 4**

During program processing, the accumulators are 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.

**Register 6: Data Block Start Address (DBA)**

When you call a data block or an extended data block using the CDB or CXDX operations, the address of DW 0 (bits 19 to 4) in the data block called is loaded into register 6. The block address list in DB 0 contains this address.

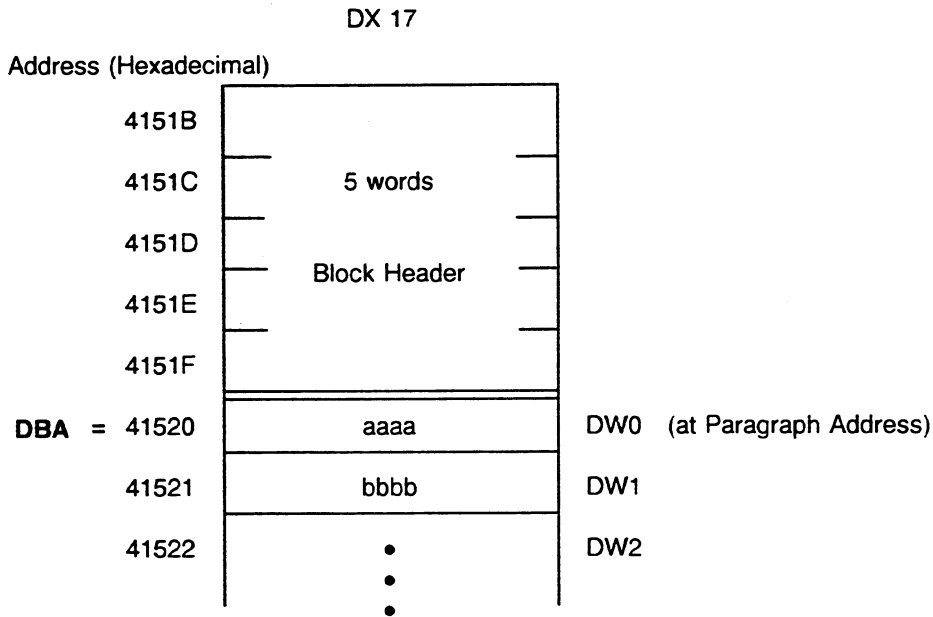
The DBA register remains the same if the following occurs:

- A jump operation (JU/JC) causes program processing to continue in a different block.
- The system program activates a different program processing level.

The DBA register changes if one of the following occurs:

- Another data block is called.
- A block that calls another block is terminated by the block end statement BE.

**Example:** Calling DX 17 and Loading the DBA Register (CX DX 17)



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 **4152** (hexadecimal).

**Note:** In the ISTACK, the address entered in the DBA register appears under the heading DB-ADD.

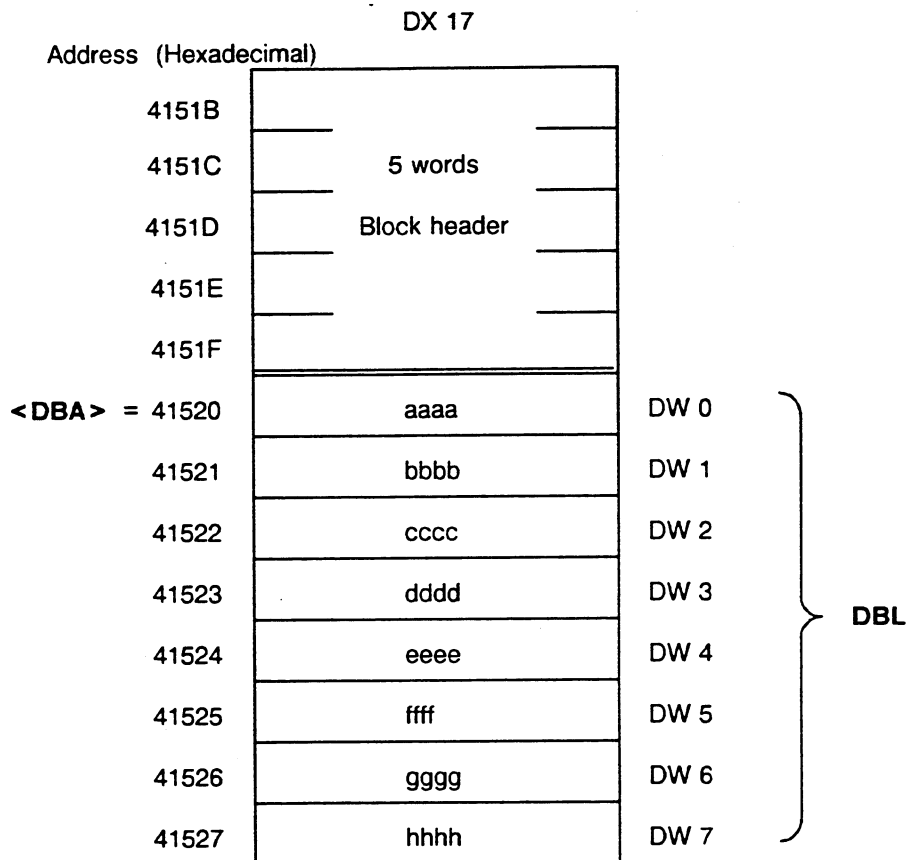
## Memory Access Using Absolute Addresses

You can access data words using the following STEP 5 operations: LDW, TDW, LDR, TDR, LDL, TDL, LDD, TDD, A Dxxx.y, O Dxxx.y, AN Dxxx.y, ON Dxxx.y, S Dxxx.y, R Dxxx.y, = Dxxx.y. You can only use these operations through DW 255. However, by manipulating the DBA register, you can use them to access data words greater than DW 255. You must also change the DBL register. Reset the DBA register to its old value before calling another block.

### Register 8: Data Block Length (DBL)

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 following is an example of loading the DBL register:

**Example:** Calling DX 17 and Loading the DBL Register (CX DX 17)



When DX 17 is called, the number of existing data words is entered in the DBL register. In this example, the DBL is 8 (DW 0 to DW 7).

**Note:** In the ISTACK, the number entered in the DBL register appears under the heading DBL-REG.



**LDI and TDI: Loading to or Transferring from a 32-Bit Memory Area Indirectly**

The following table shows the register assignment for the operations LDI and TDI for the CPU 946/947.

| Register Name (Parameter) | Register Assignment                                   |
|---------------------------|---|
| A1                        | ACCU 1 (bits 0 to 31) <sup>1)</sup>                   |
| A2                        | ACCU 2 (bits 0 to 31)                                 |
| SA                        | STEP address counter (SAC) (bits 0 to 19)             |
| BA                        | Block start address (BA) register (bits 0 through 19) |
| BR                        | Base address register (BR) (bits 0 through 19)        |

231 20

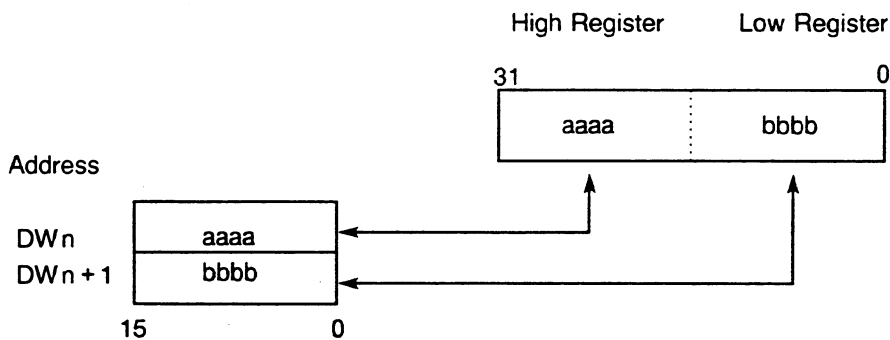
<sup>1)</sup> Loading the contents of an addressed memory register into the A1 register overwrites the address stored in ACCU 1-High/Low.

If you indicate register names other than the ones listed when using the LDI or TDI operations, the system program detects a substitution error (SUF) and calls OB 27. If **OB 27** is not programmed, the CPU goes into the STOP mode with the SUF error message.

If you reference byte addresses with the LDI or TDI operations, note the following:

- With LDI, nondefined values are written to the high bytes of the registers.
- With TDI, only the low bytes of the registers are transferred. The high bytes are lost.

**Storage of Data with the LDI or TDI Operations**



### **SA Register: STEP Address Counter (SAC)**

At the end of STEP 5 program processing, the 20-bit absolute address of the operation in the program memory to be processed next is entered in the SA register.

### **BA Register: Block Start Address (BA)**

During STEP 5 program processing, a 20-bit absolute address is entered in the BA register. This address is in the block where the call originated (i.e., the higher-order block). It is the address of the operation to be processed next.

### **BR Register: Available Base Address Register (BR)**

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.

## 9.2 Transferring Memory Blocks

You can use the TNW, TXB, and TXW operations to transfer entire memory blocks.

|     |          |  |
|-----|----------|--|
| TNW | 0 to 255 | Transfer a memory block by words (into a 16-bit memory area).    |
| TXB | -        | Transfer a memory block (from an 8-bit to a 16-bit memory area). |
| TXW | -        | Transfer a memory block (from a 16-bit to an 8-bit memory area). |

#### **NOTE**

**Do not use the TNW and TXB operations to write EPROM submodule addresses. Doing so overwrites the parity memory bits.**

For the TNW operation, the parameter specifies the length (number of words) of the area to be transferred. For the TXB and TXW operations, the length of the block (number of words) must be specified in ACCU 3 (0 to 127).

Before you execute any of these operations, you must load the end address of the source area (20 bits) into ACCU 2 and the end address of the destination area (20 bits) into ACCU 1. In each case you specify the high byte address of the source and destination areas. CPU 946/947 handles the transfer as a decrement. It begins the transmission with the highest address of the source area (i.e., end address) and ends it with the lowest address.

The entire source and destination areas must be located in the same memory area and cannot overlap. The following memory areas are distinguished by memory boundaries.

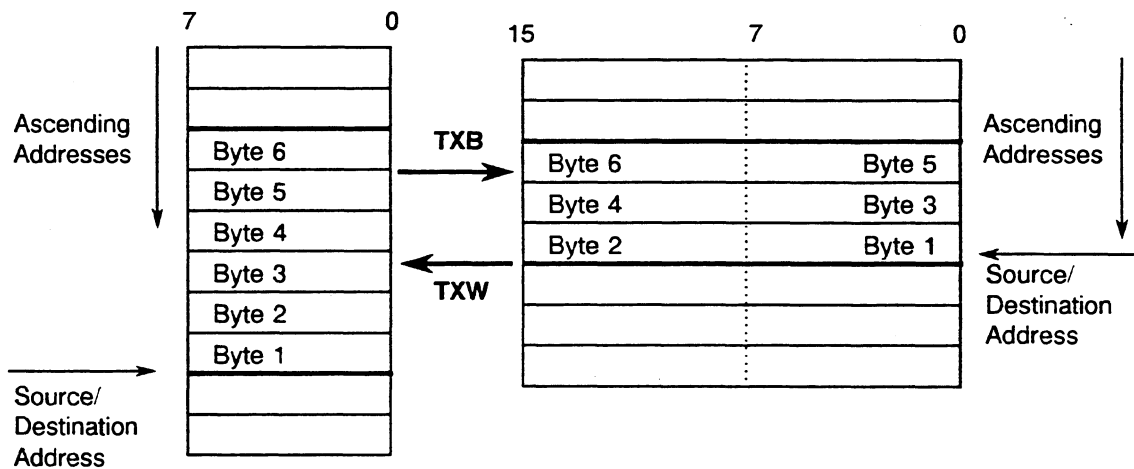
Addresses (hexadecimal)

- |                     |  |
|---------------------|--|
| 1. 0 0000 to 6 FFFF | User memory (RAM - 16 bit)                         |
| 8 0000 to D FFFF    | User memory (EPROM - 16 bit)                       |
| 2. E 8000 to E 9FFF | System RAM (16 bit) , RI/RS, timers, counters etc. |
| E B000 to E FBFF    | System RAM (16 bit) , RI/RS, timers, counters etc. |
| 3. E A000 to E AFFF | S flag (Low byte in 16-bit word)                   |
| 4. E FC00 to E FFFF | System RAM (8 bit) , flags, process image tables   |
| 5. F 0000 to F FFFF | Peripherals (8 bit)                                |

(see also Chapter 8)

If you specify incorrect area boundaries, the system program detects a transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1F01 as an error ID.

**Example: TXB and TXW between 8 and 16-Bit Memory Areas**



Transferring bytes 1 to 6 from an 8-bit to a 16-bit area

```

:L (block length in words)      :L   KH 3
:L (source address)            :L   KH EFC10
:ENT                            :ENT
:L (destination address)       :L   KH EF008
:TXB                            :TXB
    
```

Transferring Bytes 1 to 6 from a 16-bit to an 8-bit Area

```

:L (block length in words)      :L   KH 3
:L (source address)            :L   KH EF008
:ENT                            :ENT
:L (destination address)       :L   KH EFC10
:TXW                            :TXW
    
```

## Memory Access Using Absolute Addresses

---

### Interruptions during the TNW, TXB, and TXW Operations

A power failure (NAU) or a timeout error (QVZ) affects the execution of these operations, as follows:

- a) NAU - If you perform a warm restart after a power failure, the operation in question does not continue. Instead, the operation is processed from the beginning.
- b) QVZ - If the QVZ is acknowledged in an error OB, program processing continues with the next operation.

### 9.3 Operations with the Base Address Register

You can use the MBR and ABR operations to reload or modify the contents of the BR register.

| Operation | Parameter                                       | Explanation   |
|-----------|---|---|
| MBR       | Constants<br>0 to FFFFF<br>(hexadecimal)        | Load the BR register with a 20-bit address.               |
| ABR       | Constants<br>- 32768 to<br>+ 32767<br>(decimal) | Add a 16-bit constant to the contents of the BR register. |

The following operations make addressing of system data easy by providing you with indirect memory access via the base register (BR, 20 bits). You do not have to use the accumulators to load an address.

| Operation | Parameter                                    | Explanation   |
|-----------|--|---|
| LRW       | Offset<br>- 32768 to<br>+ 32767<br>(decimal) | Add the offset to the contents of the BR register and load the contents of the memory register addressed this way into ACCU 1-Low (word). <sup>1)</sup>   |
| LRD       | Offset<br>- 32768 to<br>+ 32767              | Add the offset to the contents of the BR register and load the contents of the memory register n addressed this way into ACCU 1-High. Load the contents of memory register n + 1 into ACCU 1-Low (double word). <sup>2)</sup> |

<sup>1)</sup> ACCU 2 = ACCU 1  
ACCU 1-High = 0000H

<sup>2)</sup> ACCU 2 = ACCU 1

| Operation | Parameter                       | Explanation   |
|-----------|---------------------------------|---|
| TRW       | Offset<br>- 32768 to<br>+ 32767 | Add the offset to the contents of the BR register and load the contents of ACCU 1-Low into the memory register addressed this way (word).   |
| TRD       | Offset<br>- 32768 to<br>+ 32767 | Add the offset to the contents of the BR register and load the contents of ACCU 1-High into the memory register n addressed this way. Load the contents of ACCU 1-Low into memory register n + 1 (double word). |

For all four operations, the BR register is **not changed** after the operation is executed.

A value between - 32 K and +32 K -1 is permitted as an offset.

If the calculated address of the memory register is not between "00000" and "EFBFF," the system program detects a load or transfer error and calls **OB 32**. ACCU 1 contains 1A01 as an error ID. If OB 32 is not programmed, the CPU goes into the STOP mode with the error message TLAF.

## 9.4 Operations for Transferring the Contents of One Register to Another

To transfer data between register 1, SAC and the BR register, the following operations are available:

| Operation | Parameter | Explanation   |
|-----------|-----------|---|
| MAS       | -         | Transfer the contents of ACCU 1 <sup>1)</sup> to the STEP address counter (SA register) (20 bits).                              |
| MAB       | -         | Transfer the contents of ACCU 1 <sup>1)</sup> to the base address register (BR register) (20 bits).                             |
| MSA       | -         | Transfer the contents of the STEP address counter (SA register) (20 bits) to ACCU 1 <sup>2)</sup> .                             |
| MSB       | -         | Transfer the contents of the STEP address counter (SA register) (20 bits) to the base address register (BR register) (20 bits). |
| MBA       | -         | Transfer the contents of the base address register (BR register) (20 bits) to ACCU 1 <sup>2)</sup> .                            |
| MBS       | -         | Transfer the contents of the base address register (BR register) (20 bits) to the STEP address counter (SA register) (20 bits). |

1) If ACCU 1 is the **source** of the transfer operation, bits 31 to 20 are not changed.

2) If ACCU 1 is the **destination** of the transfer operation, bits 31 to 20 are set to zero.

## 9.5 Accessing the Global Memory

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 reenable the memory area*.

The TSG operation enables testing and setting of "occupied" registers.

| Operation | Parameter             | Explanation  |
|-----------|-----------------------|--|
| TSG       | - 32768 to<br>+ 32767 | Test and set the "occupied" register addressed by the sum of the BR register and constant. |

The absolute address must be between F0000 and FFFFF (hexadecimal code). If it is not, the system program detects a transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.

The low byte of the word addressed by the sum of the BR register and constant is used as the "occupied" register. If the low byte contains 0, the TSG operation enters the slot ID of the CPU in the "occupied" register.

Testing (reading) and possible occupancy (writing) form a program unit that cannot be interrupted.

You can evaluate the result of the test using condition codes CC 1 and CC 0:

| CC 1 | CC 0 | Explanation   |
|------|------|---|
| 0    | 0    | The "occupied" register contained 0. The CPU entered 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 memory area must use the TSG operation.**

For related information, see the explanations of the SED and SEE (disable/enable semaphore) operations in section 3.3.

### Load and Transfer Operations for the Global Memory Organized in Bytes

| Operation | Parameter            | Explanation  |
|-----------|----------------------|--|
| LY GB     | -32768 to<br>+ 32767 | Load the byte addressed by the sum of the BR register and constant into ACCU 1-LL. <sup>1), 2)</sup>     |
| LY GW     | -32768 to<br>+ 32767 | Load the word addressed by the sum of the BR register and constant into ACCU 1-Low. <sup>1), 3)</sup>    |
| LY GD     | -32768 to<br>+ 32767 | Load the double word addressed by the sum of the BR register and constant into ACCU 1. <sup>1)</sup>     |
| TY GB     | -32768 to<br>+ 32767 | Transfer the contents of ACCU 1-LL to the byte addressed by the sum of the BR register and constant.     |
| TY GW     | -32768 to<br>+ 32767 | Transfer the contents of ACCU 1-Low to the word addressed by the sum of the BR register and constant.    |
| TY GD     | -32768 to<br>+ 32767 | Transfer the contents of ACCU 1 to the double word addressed by the sum of the BR register and constant. |

<sup>1)</sup> ACCU 2 = ACCU 1

<sup>2)</sup> ACCU 1-LH and ACCU 1-High are set to 0.

<sup>3)</sup> ACCU 1-High is set to 0.

The range of absolute addresses for the load and transfer operations for the global memory organized in bytes.

- between F0000 and FFFFF (LB GB and TB GB),
- between F0000 and FFFFE (LB GW and TB GW),
- between F0000 and FFFFC (LB GD and TB GD).

If the absolute addresses are not in the range shown, the system program detects a load/transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.



**Load and Transfer Operations for the Global Memory Organized in Words**

| Operation | Parameter            | Explanation  |
|-----------|----------------------|--|
| LW GW     | -32768 to<br>+ 32767 | Load the word addressed by the sum of the BR register and constant into ACCU 1-Low. <sup>1), 2)</sup>    |
| LW GD     | -32768 to<br>+ 32767 | Load the double word addressed by the sum of the BR register and constant into ACCU 1. <sup>1)</sup>     |
| TW GW     | -32768 to<br>+ 32767 | Transfer the contents of ACCU 1-Low to the word addressed by the sum of the BR register and constant.    |
| TW GD     | -32768 to<br>+ 32767 | Transfer the contents of ACCU 1 to the double word addressed by the sum of the BR register and constant. |

<sup>1)</sup> ACCU 2 = ACCU 1  
<sup>2)</sup> ACCU 1-High is set to 0.

If the absolute addresses are not in the range between F0000 and FFFFF (with LW GW and TW GW) or between F0000 and FFFFE (with LW GD and TW GD), the system program detects a load/transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.

**9.5.1 Accessing the Dual-Port RAM Memory**

Between the addresses FF400 and FFBFF, the global memory area has a window for accessing one of a maximum of 256 memory areas called dual-port RAM pages. One dual-port RAM page can occupy a maximum of 2K addresses and can be organized in *either bytes or words*. Before access to the dual-port RAM area, one of the 256 pages must be selected by entering its number in the select register.

You can access dual-port RAM pages using absolute addresses. This section describes the operations you can use to do this. The absolute address is the sum of the contents of the BR register and the constants contained in the operation ( - 2048 to + 2047).

Before you can access the dual-port RAM area (load/transfer), you must select one of the 256 dual-port RAM pages. Do this by putting the number of the dual-port RAM page that you want to open into ACCU 1-Low. Use the ACR operation to enter this number into the dual-port RAM register. All dual-port RAM operations that follow ACR write the contents of the dual-port RAM register into the select register before dual-port RAM access. The procedure of writing to the select register and then accessing the dual-port RAM area cannot be interrupted.

The dual-port RAM register is maintained under the following circumstances:

- A jump operation (JU/JC) causes program processing to continue in a different block.
- The system program activates a different program processing level.

## Memory Access Using Absolute Addresses

---

### Opening a Dual-Port RAM Page

| Operation | Parameter | Explanation  |
|-----------|-----------|--|
| ACR       |           | Open the dual-port RAM page whose number is located in ACCU 1-LL . |

permissible values are 0 to 255

The dual-port RAM page number must be between 0 and 255. If it is not, the system program detects a substitution error (SUF) and calls **OB 27**.

### Testing and Setting an "Occupied" Register in the Dual-Port RAM 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 reenable the memory area.*

The TSC operation supports testing and setting of an "occupied" register.

| Operation | Parameter           | Explanation   |
|-----------|---------------------|---|
| TSC       | - 2048 to<br>+ 2047 | Test and set the "occupied" register addressed by the sum of the BR register and constant from the opened dual-port RAM page. |

The absolute address must be between FF400 and FFBF (hexadecimal code). If it is not, the system program detects a load/transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.

The low byte of the word addressed by the sum of the BR register and 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 occupancy (writing) form a program unit that cannot be interrupted.

You can evaluate the result of the test using condition codes CC 1 and CC 0.

| CC 1 | CC 0 | Explanation   |
|------|------|---|
| 0    | 0    | The "occupied" register contained 0. The CPU entered 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 memory area must use the TSC operation.**

For related information, see the explanations of the SED and SEE (disable/enable semaphore) operations in section 3.3.

**Load and Transfer Operations for the Dual-Port RAM Memory Organized in Bytes**

| Operation | Parameter      | Explanation   |
|-----------|----------------|---|
| LY CB     | -2048 to +2047 | Load the byte addressed by the sum of the BR register and constant from the opened dual-port RAM page into ACCU 1-LL. <sup>1), 2)</sup>   |
| LY CW     | -2048 to +2047 | Load the word addressed by the sum of the BR register and constant from the opened dual-port RAM page into ACCU 1-Low. <sup>1), 3)</sup>  |
| LY CD     | -2048 to +2047 | Load the double word addressed by the sum of the BR register and constant from the opened dual-port RAM page into ACCU 1. <sup>1)</sup>   |
| TY CB     | -2048 to +2047 | Transfer the contents of ACCU 1-LL to the byte addressed by the sum of the BR register and constant in the opened dual-port RAM page.     |
| TY CW     | -2048 to +2047 | Transfer the contents of ACCU 1-Low to the word addressed by the sum of the BR register and constant in the opened dual-port RAM page.    |
| TY CD     | -2048 to +2047 | Transfer the contents of ACCU 1 to the double word addressed by the sum of the BR register and constant in the opened dual-port RAM page. |

<sup>1)</sup> ACCU 2 = ACCU 1

<sup>2)</sup> ACCU 1-LH and ACCU 1-High are set to 0.

<sup>3)</sup> ACCU 1-High is set to 0.

## Memory Access Using Absolute Addresses

The range of absolute addresses for the load and transfer operations for the dual-port RAM memory organized in bytes.

- between FF400 and FFBFF (LY CB and TY CB),
- between FF400 and FFBFE (LY CW and TY CW),
- between FF400 and FFBFC (LY CD and TY CD).

If the absolute addresses are not in the range shown below, the system program detects a load/transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.

### Load and Transfer Operations for the Dual-Port RAM Memory Organized in Words

| Operation | Parameter           | Explanation   |
|-----------|---------------------|---|
| LW CW     | -2048 bis<br>+ 2047 | Load the word addressed by the sum of the BR register and constant from the opened dual-port RAM page into ACCU 1-Low. <sup>1), 2)</sup>  |
| LW CD     | -2048 bis<br>+ 2047 | Load the double word addressed by the sum of the BR register and constant from the opened dual-port RAM page into ACCU 1. <sup>1)</sup>   |
| TW CW     | -2048 bis<br>+ 2047 | Transfer the contents of ACCU 1-Low to the word addressed by the sum of the BR register and constant in the opened dual-port RAM page.    |
| TW CD     | -2048 bis<br>+ 2047 | Transfer the contents of ACCU 1 to the double word addressed by the sum of the BR register and constant in the opened dual-port RAM page. |

<sup>1)</sup> ACCU 2 = ACCU 1

<sup>2)</sup> ACCU 1-High is set to 0.

If the absolute addresses are not between FF400 and FFBFF (with LW CW and TW CW) or FF400 and FFBFE (with LW CD, TW CD), the system program detects a load/transfer error (TLAF) and calls **OB 32**. ACCU 1 contains 1A01 as an error ID.

## 9.6 Set Operations in the RS/RT Area

You can use the following system operations to set or reset individual bits in the system data registers (RS 0 to RS 255 and RT 0 to RT 255) (see section 8.2.4).

| Operation | Parameter        | Explanation  |
|-----------|------------------|--|
| SU        | RS 0.0 to 255.15 | Set (unconditionally)<br>A bit in the system data area (RS area)   |
|           | RT 0.0 to 255.15 | A bit in the extended system data area (RT area)                   |
| RU        | RS 0.0 to 255.15 | Reset (unconditionally)<br>A bit in the system data area (RS area) |
|           | RT 0.0 to 255.15 | A bit in the extended system data area (RT area)                   |

## Chapter 10

# Multiprocessing with Communications Processors (CPs)

### 10.1 Notes

You can operate a maximum of four CPUs simultaneously in the S5-155U central controller to handle complex and technologically separable automation tasks. Each CPU processes its program independently.

In addition to one CPU 946/947, you can use the following CPUs in various combinations for multiprocessor operation:

- A second CPU 946/947 with a maximum of two 355 memory modules. The maximum configuration per CPU 946/947 occupies five slots.
- CPU 928B. This CPU occupies two slots.
- CPU 928. This CPU occupies two slots.
- CPU 922 (R processor). This CPU occupies one slot.
- CPU 920 (M processor). This CPU occupies one slot.

For more information on configuration, see the S5-155U Central Controller Housing Hardware and Installation Guide, "Possible Configurations" chapter.

**Using the S5-155U as a multiprocessing device could be practical under any of the following circumstances:**

- When a particular part of your system has to be processed especially fast, separate the appropriate program part from the total program and let a separate, fast CPU process it.
- When your system consists of several parts that you can separate easily and submit to open/closed-loop control independently, let CPU 1 process system part 1, CPU 2 process system part 2, etc.

For more information on multiprocessing, see the description "Multiprocessing in the S5-135U and S5-155U". It explains the step-by-step procedure for starting up your multiprocessor and provides useful information for operation. It describes a few typical error situations and makes reference to some possible causes of error.

**NOTE**

**The S5-155U is set up for multiprocessor signaling operation as soon as you plug a Coordinator 923C into the central controller housing, whether you are operating the coordinator with one or more CPUs.**

**For successful multiprocessor operation, you must program extended data block DX 0 and set CPU 946/947 to the 155U controller mode in it.**

If the S5-155U is in the 150U controller mode and you have plugged in a Coordinator 923C, the system program detects an error. This is true even if you plugged in the coordinator only to use the programmer multiplexer. CPU 946/947 goes into the STOP mode.

Coordinator 923C coordinates data exchange between individual CPUs. To do this, the coordinator must know which slots have CPUs. It must know the number of CPUs to which it must allocate S5 bus-access time slices sequentially.

**NOTE**

**Make the appropriate setting on the coordinator for the number of CPUs used (see the 923 C Coordinator Instructions).**

If you set the number of CPUs to 3 on the coordinator, it distributes the bus-enable signal sequentially for the CPUs that occupy the first three CPU slots from left to right in the central controller housing (numbers 17, 51, and 91)(see the S5-155U Central Controller Housing Hardware and Installation Guide). The fourth slot (number 99) is not handled by the coordinator in this case. If you plug a CPU into that slot, it cannot access the S5 bus.

**NOTE**

**Plug in Coordinator 923C and all CPUs from left to right without leaving any intermediate CPU slots empty.**

**NOTE**

**If you are using multiprocessing, you must program DB 1 for each CPU used. DB 1 contains a list of digital inputs and outputs and interprocessor communication flag inputs and outputs that are assigned to each CPU (see section 10.3.1).**

## 10.2 Data Exchange between CPUs

This section describes how CPUs can exchange data with other CPUs or with CPs during the operating cycle. Various aids are available to help you implement different types of exchange as follows:

**Interprocessor communication flags** - For cyclic exchange of binary data between CPUs or between a CPU and CPs (see section 10.2.1).

**Special functions for multiprocessing** - For exchange of large amounts of data (e.g., entire data blocks) between individual CPUs (see section 10.2.2).

**Handling blocks** - For communication with intelligent input/output modules (IPs) and with CPs (see section 10.2.3). These handling blocks must be ordered separately.

**Semaphores** - For transferring large data blocks and ensuring that other CPUs do not interfere with the transmission (see section 3.3).

### 10.2.1 Interprocessor Communication Flags

Interprocessor communication (IPC) flags are available for cyclic exchange of binary data. They are used mainly for transmitting information byte by byte. Data is transferred as follows:

CPU(s) ↔ CPU(s)

CPU(s) ↔ CP(s)

The system program transfers IPC flags once per cycle. For data transfer between several CPUs, the IPC flags are buffered physically in the coordinator.

IPC flags are flag bytes that are transferred. You define them in DB 1 for each CPU as input or output IPC flags.

If you define flag byte FY 50 in CPU 1 as an **output** IPC flag, the system program transfers its signal status to the CPU in which you defined flag byte FY 50 as an **input** IPC flag. The system program makes this transfer cyclically via the coordinator.

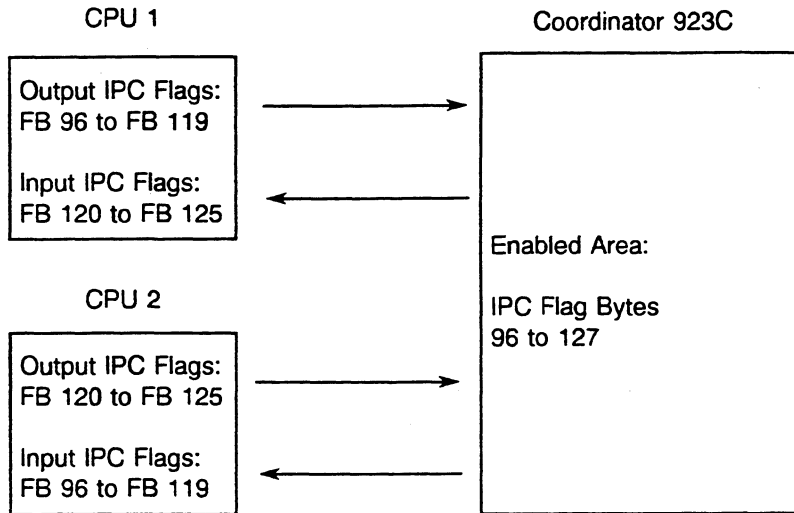
The memory area for the IPC flags in the coordinator and the CPs includes the addresses **FF200H to FF2FFH**. You can allocate this area as needed in groups of 32 bytes to the various modules (via jumper settings on the coordinator and the CPs). You have 256 IPC flag bytes available for each CPU.

### Data Exchange between CPUs

The coordinator handles data exchange between individual CPUs (CPU 946/947, CPU 928, CPU 922, and CPU 920). The CPUs read their flag bytes defined in DB 1 as input IPC flags from the coordinator; they write their flag bytes defined as output IPCs to the coordinator.

You must enable the necessary number of IPC flags on the coordinator. By setting jumpers, you can divide the maximum of 256 IPC flag bytes into groups of 32 bytes (eight groups). For more information see the 923 C Coordinator Instructions.

Example:



#### NOTE

The only flag bytes that you can use as IPC flags are the ones that you enabled by removing jumpers on the coordinator.

A flag byte that is defined in DB 1 of one or more CPUs as an input IPC flag byte must be defined as an output IPC flag byte on one other CPU. An IPC output byte is only allowed on one CPU, but this IPC byte may be used as an IPC flag input in all other CPUs in the rack.

If you have flag bytes that you have not defined as IPC flags in a CPU, you can use them as normal flags.

In DB1, indicate only the number of IPC flags that you actually need: the smaller the number of IPC flags, the shorter the transfer time.

**S flags cannot be used as IPC flags.**

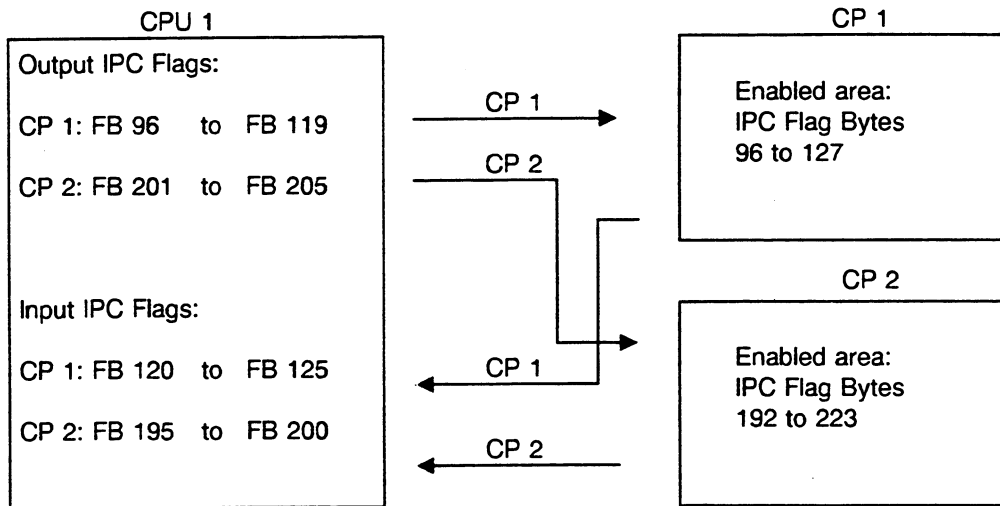


**Data Exchange between CPUs and CPs with IPC flags**

If you want to exchange data between *one* CPU and *one* CP, you must enable the necessary number of IPC flags on the CP. You have 256 IPC flag bytes available that you can divide into groups of 32 bytes each. These 256 bytes are the total number of bytes available for data exchange between CPUs and between CPUs and CPs.

If you want to transfer data from *one* CPU to *several* CPs, the areas you enable in the CPs must not overlap, otherwise the same address is assigned twice.

Example:



If you want to use IPC flags *simultaneously* in the coordinator and in one or more CPs, you must also prevent double addressing as follows:

- Divide the IPC flags among the coordinator and the CPs in groups of 32 bytes each. Remove jumpers on the coordinator to mask the IPC flag bytes that you want to use in the CP (see the instructions for the coordinator you use).

You can define a specific flag byte as an output IPC flag in one CPU only. However, you can define a specific flag byte as an input IPC flag in several CPUs.

**Transmitting IPC Flags in Multiprocessor Operation**

At the end of the cycle, the CPU transmits the IPC flags specified in DB 1 when the coordinator signals the CPU that it can access the S5 bus.

The coordinator allocates the bus enable signal to each CPU in sequence (see below). When a CPU has access to the S5 bus, it can transmit only one byte. Because of this interleaved transmission, related pieces (byte groups) of IPC flag information can be separated and subsequently processed with old or incorrect values.

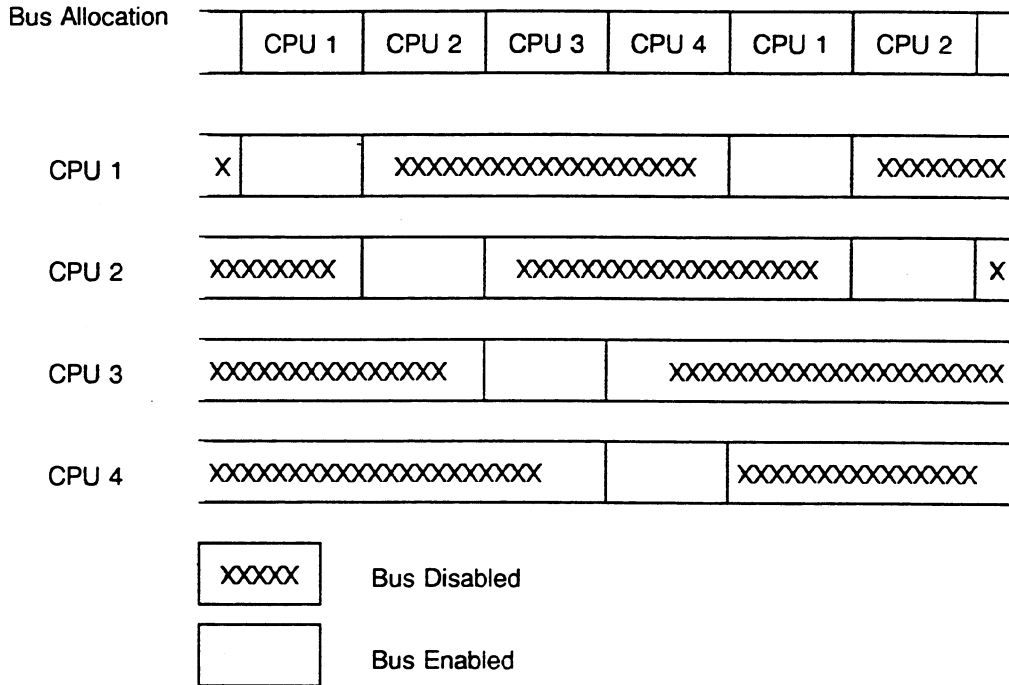


Fig. 10.1 Bus Allocation in Multiprocessor Operation

If you want to transfer information that takes up more than one byte, you can prevent corruption of data by setting a parameter in extended data block DX 0. This parameter uses semaphores to ensure that all IPC flags specified in DB 1 are transferred **in groups** (see Chapter 7). While one CPU is transmitting IPC flags, another CPU cannot interrupt it. Because the next CPU has to wait to transmit its data, cyclic program processing is delayed accordingly. Under certain circumstances, the setting you make in DX 0 can increase the cycle time considerably.

### **10.2.2 Multiprocessor Communication**

You can use special function organization blocks OB 200 and OB 202 to OB 205 in multiprocessing to transfer large amounts of data (e.g., entire data blocks) between CPUs. Such a transfer can take place using Coordinator 923C. The data is buffered in the coordinator. These OBs function are:

**OB 200 (Initialize)** - OB 200 sets up a memory in Coordinator 923C. This memory is a buffer for the data fields that are transferred.

**OB 202 (Send)** - OB 202 transfers a data field to the buffer of Coordinator 923C and indicates how many data fields can still be sent.

**OB 203 (Send Test)** - OB 203 determines the number of available memory blocks in the buffer of Coordinator 923C.

**OB 204 (Receive)** - OB 204 transfers a data field from the buffer of Coordinator 923C and indicates how many data fields can still be received.

**OB 205 (Receive Test)** - OB 205 determines the number of occupied memory blocks in the buffer of Coordinator 923C.

For a detailed user's guide to these special function organization blocks, see Multiprocessor Communication: S5-135U, CPU 922 (R Processor) and CPU 928; S5-155U, CPU 946/947 to be found in the S5-135U and S5-155U manuals.

### **10.2.3 Handling Blocks**

Handling blocks are standard function blocks that control data traffic between a CPU and a CP for a relatively large volume of data via the dual-port RAM area in single and multiprocessor operation. Use handling blocks when you have to transfer data, parameters, or open-loop control information to or from CPs.

You can order handling blocks as a software product on diskette (see Standard Function Blocks and Driver Software for Programmable Controllers of the U- Range Catalog ST 57).

### **10.3 Allocation of Peripherals**

The global peripheral area of all CPUs includes the addresses **F0000H to FFFFFH**. The peripheral modules are addressed in this area. The IPC flags and the dual-port RAM pages are also in this area. All CPUs have read/write access to this peripheral area. The coordinator coordinates access of individual CPUs to the peripheral area. The coordinator also makes sure that only one CPU at a time has access to the area.

The coordinator gives each CPU a signal in sequence, granting it access to the peripheral bus for a specific period of time. If the CPU does not finish its data transfer within this time, the transfer is broken off before the next byte is read or written. Data transfer is resumed at this point when the CPU has access to the peripheral bus again.

#### **10.3.1 Data Block DB 1**

For multiprocessing, you must program DB 1 for each CPU. This establishes the inputs, outputs, and IPC flag inputs and outputs with which each CPU works.

**NOTE**

**The system program considers only the inputs and outputs defined in DB 1 when it updates the process image.**

#### **Inputting or Changing DB 1**

You can input or change DB 1 as follows:

- 1) on-line via a programmer when the CPU is in the STOP or RUN mode, if the 355 memory module is equipped with RAM submodules.
- 2) by programming the EPROM submodules.

**NOTE**

**The CPU accepts the entered or changed DB 1 only after a cold restart.**

DB 1 can be assigned parameters via a menu on the PG or directly in hexadecimal. See the following example:

**Programming DB 1 Using Softkeys:**  
 (the numbers are the byte addresses of the modules)

| DB 1                  |   |     |     |      |     |    |     |   |   |   |
|-----------------------|---|-----|-----|------|-----|----|-----|---|---|---|
| DIGITAL INPUTS        | , | 0,  | 1,  | 2,   | 3,  | 7, | 10, | , | , | , |
| DIGITAL OUTPUTS       | , | 0,  | 2,  | 4,   | 12, | ,  | ,   | , | , | , |
| IPC FLAG INPUTS       | , | 50, | 51, | 60,  | ,   | ,  | ,   | , | , | , |
| IPC FLAG OUTPUTS      |   | 70, | 72, | 100, | ,   | ,  | ,   | , | , | , |
| TIMER BLOCK LENGTH 1) | , |     |     |      |     |    |     |   |   |   |

1) Attention: Entry of the time block length in DB 1 is ignored by CPU 946/947. The default value (256 timers) remains valid. You can make a setting in DX 0 to indicate the number of timers you want to update (see Chapter 7).

**Creating DB 1 in Hexadecimal Form:**

- Assign the start ID "MASK 01" for DB 1 to DW 0, DW 1, and DW 2 as follows:

DW 0: KH = 4D41 ('M' 'A')  
 DW 1: KH = 534B ('S' 'K')  
 DW 2: KH = 3031 ('0' '1')

- Beginning with DW 3, indicate the individual operand areas.

Enter a specific ID for each operand area.  
 The possible ID words are as follows:

|                              |           |
|------------------------------|-----------|
| ID word for digital inputs   | KH = DE00 |
| ID word for digital outputs  | KH = DA00 |
| ID word for IPC flag inputs  | KH = CE00 |
| ID word for IPC flag outputs | KH = CA00 |

After each ID word, use fixed-point format to list the numbers of the inputs and outputs used.

- Assign the value KH = EEEE to the last data word to designate it as the end ID for DB 1.

**NOTE**

You can make the DB1 entries in any order. The system program updates the process image of your last entry first.

Multiple entries of the same bytes (e.g., for test purposes) are possible. The system program makes multiple updates of the process image tables of bytes that are entered more than once.

You must enter the end ID KH = EEEE as the last entry in DB 1.

### Example for creating DB 1 manually

```
0 :      KH = 4D41;          DW 0-2:
1 :      KH = 534B;          Start ID
2 :      KH = 3031;          for DB 1
3 :      KH = DE00;          ID word for digital inputs
4 :      KF = +00000;        Input byte 0
5 :      KF = +00001;        Input byte 1
6 :      KF = +00002;        Input byte 2
7 :      KF = +00003;        Input byte 3
8 :      KF = +00007;        .
9 :      KF = +00010;        .
10 :     KH = DA00;          ID word for digital outputs
11 :     KF = +00000;        digital outputs 0
12 :     KF = +00002;        digital outputs 2
13 :     KF = +00004;        .
14 :     KF = +00012;        .
15 :     KH = CE00;          ID word for IPC flag inputs
16 :     KF = +00050;        Flag byte 50
17 :     KF = +00051;        .
18 :     KF = +00060;        .
19 :     KH = CA00;          ID word for IPC flag outputs
20 :     KF = 00070;        Flag byte 70
21 :     KF = +00072;        .
22 :     KF = +00100;        .
23 :     KH = EEEE;          End ID
```

The system program accepts DB 1 during a cold restart. The system program checks to see if the inputs and outputs or IPC flags indicated in DB 1 exist in their corresponding modules. If they are not present there, a DB 1 error causes the CPU to go into the smooth STOP mode and the STOP LED flashes slowly. The CPU no longer processes your program.

After you program DB 1 and the CPU accepts it during a cold restart, the following rules apply:

- Only the inputs and outputs indicated in DB 1 can access peripheral modules *via the process image tables* (L IB, L IW, L ID, L QB, L QW, and L QD or T IB, T IW, T ID, T QB, T QW, and T QD operations and logic operations with inputs and outputs).
- You can load peripheral bytes directly by *bypassing the process image* using the L PY, L PW, L OB, and L OW operations for all acknowledging inputs, regardless of entries in DB 1.
- You can transfer directly to bytes 0 to 127 using the T PY and T PW operations only for the outputs indicated in DB 1. This is because the process image is also written to during direct transfer. You can transfer directly to byte addresses greater than 127 regardless of entries in DB 1. You can also transfer directly to byte addresses of the extended peripheral area (T OB and T OW) regardless of entries in DB 1.

## 10.4 Start-Up Procedure for Multiprocessing

You can start the coordinator for multiprocessing in one of the following ways:

- Make sure the RUN/STOP switch of each plugged CPU is in the RUN position. The RUN/STOP switch on the coordinator is in the STOP position.
- Move the RUN/STOP switch on the coordinator from STOP to RUN.

(Starting the S5-155U in multiprocessor operation simply by starting the coordinator is possible only if the coordinator itself caused the controller to enter the STOP mode and the problem in the coordinator has been corrected.)

or

- Make sure the RUN/STOP switch of each plugged CPU is in the RUN position and on the coordinator in the RUN position.
- Use the START programmer function to start the CPU that caused the controller to enter the STOP mode. After the problem has been corrected in that CPU, choose the restart type you want.

The restart type that each CPU now uses depends on what took place while the CPU was in the STOP mode. Some CPUs need a manual warm restart, others, a cold restart.

If a CPU was not used in that time, execute a manual warm restart.

### NOTE

**Through various restart types, incorrect signal statuses can be transferred from one CPU to another via the IPC flags. This could happen if the controller was in cycle prior to entering the STOP mode. Prevent this by programming OB 20, OB 21 and OB 22 appropriately.**

You can call special function organization block **OB 223** to find out if the restart types of all CPUs participating in multiprocessing are the same (see Chapter 6).

When power is shut off and then restored, the coordinator starts automatically. In this case, the CPUs execute an automatic warm restart or an automatic cold restart, depending on the presetting in DX 0 (see Chapter 7).

Start-up of the individual CPUs in multiprocessor operation is **synchronized**. Each CPU waits until all others have ended their initialization phase. Then they begin their cycle simultaneously. However, you can use a setting in DX 0 to cancel start-up synchronization.

## 10.5 Test Operation

Initiate test operation as follows:

- Enable the function TEST OPERATION on the coordinator by using a DIP switch selection (see the *923 C Coordinator Instructions*).
- Move the RUN/STOP switch on the coordinator from STOP to TEST. The BASP LED goes out.
- Perform the type of restart for the CPUs you want to test.

### Special Features of Test Operation

In test operation, you can run CPUs individually or in any combination. CPUs in the STOP mode cannot disable the entire controller.

Start-up of individual CPUs is *not* synchronized in test operation. The CPUs begin their cyclic operation at different times according to the length of the start-up OBs (OB 20, OB 21, and OB 22).

If an error occurs in one CPU during test operation, only that CPU goes into the STOP mode. The error does not affect the other CPUs.

#### NOTE

**Output of the "BASP" signal is suppressed for all CPUs.  
The digital peripheral outputs are not disabled when an  
error occurs.**



#### Warning

After start-up is completed, move DIP switch on the coordinator from the "ON" to the "OFF" position to make sure the test operation is deactivated. This prevents improper operation that could result in dangerous system conditions.



**Summary: Starting Up Your Programmable Controller in Multiprocessor Operation**

Start up your programmable controller using the following procedure:

- On the coordinator, set the number of CPUs to be used.  
Enable the IPC flags on the coordinator.
- Plug the CPUs into the central controller housing, leaving no intermediate CPU slots empty.
- Turn on the main power.
- Make sure the RUN/STOP switch on the coordinator is in the STOP position.
- Perform an overall reset on all CPUs.
- Load your program into the CPUs or plug in the EPROM submodules.
- Perform a cold restart on all CPUs.
- Move the RUN/STOP switch on the coordinator to the RUN position.

## Chapter 11

### Testing Aids: Online Programmer Functions

This chapter describes some special features of online programmer functions as they are related to the CPU 946/947. Online programmer functions are an important aid to testing your user program. See the appropriate programmer manual for detailed information on how to operate the programmer and how to use these functions.

You can use the programmer to ask for access to on-line programmer functions. The system program executes these functions at defined points (system checkpoints) in programmable controller processing. The CPU 946/947 has four different checkpoints. Specific online programmer functions are assigned to each of these checkpoints.

#### **Checkpoint "Stop"**

You can access online programmer functions that are permissible only in the STOP mode at checkpoint "Stop." These functions include the following: START, OVERALL RESET, COMPRESS MEMORY in the STOP mode.

The "Stop" checkpoint is located immediately before OB 39 is called (the block for communication cyclic processing in the smooth STOP mode).

#### **Checkpoint "Cycle"**

Online programmer functions that you want to execute during cyclic program processing are called at checkpoint "Cycle." These functions include the following: COMPRESS MEMORY in the RUN mode, STOP and STATUS. The "Cycle" checkpoint is located immediately before the process image input table (PII) is updated. At this location, the system program has not updated the PII yet.

#### **Checkpoint "Test"**

Online programmer functions that you want to execute as soon as the following stopping point has been reached are called at checkpoint "Test" (see section 11.9).

#### **Checkpoint "General Functions"**

This checkpoint exists in both the STOP and the RUN mode. Online programmer functions that can be executed in all operating modes of the programmable controller are called at checkpoint "General Functions." These functions include the following: OUTPUT BLOCK, DELETE BLOCK, STATUS VARIABLES. In the STOP mode, this checkpoint is located immediately before OB 39 is called (the block for communication cyclic processing in the smooth STOP mode). During cyclic program processing, the checkpoint is located between updating the process image input and calling OB 1. At this point, the system program has already updated the PII, but has not entered the block for cyclic program processing in the RUN mode.

### 11.1 DIRECTORY

You can call the DIRECTORY programmer function to output a list of all programmed blocks or all blocks of a specific type. With the CPU 946/947, the programmer also displays all system program blocks.

This function is possible in the following modes: RUN, smooth STOP, hard STOP.

You can also call this function within the PROGRAM TEST function.

### 11.2 DELETE

You can call the DELETE programmer function to perform an overall reset. This programmer function executes an overall reset unconditionally (see section 4.2).

You can execute this function only in the smooth and hard STOP modes.

After an overall reset, a cold restart is required.

### 11.3 START and STOP

When you use the START and STOP programmer functions, operating the programmer corresponds to manual operation.

You can put the programmable controller into the STOP mode by calling the PG STOP function while the controller is in the RUN mode. The condition of the LEDs on the CPU to which the programmer is connected is as follows:

LED STOP: On  
LED BASP: On

PGSTP is marked in the control bit display. In multiprocessor operation, the HALT control bit is set for the other CPUs.

In the smooth STOP mode, you can use the START function to perform a cold or warm restart, as follows: Press the <START> softkey, select warm or cold restart. After you follow this procedure, the CPU reacts differently for single and multiprocessor operation, as follows: Single processor operation - The CPU goes into the RUN mode. Multiprocessor operation - The restart type is registered initially (the NEUDF or WIEDF control bit is set). However, the CPU stays in the smooth STOP mode until all CPUs are initialized for multiprocessing. At that point, all CPUs begin cyclic program processing simultaneously. This corresponds to operating Coordinator 923C (i.e., setting the RUN/STOP switch to the RUN position).

You can call the START programmer function in multiprocessing to select the restart type you want for all the CPUs you are using. After that, you can start the programmable controller at the last CPU.

#### 11.4 COMPRESS MEMORY

You can call the COMPRESS MEMORY programmer function to shift all valid blocks in the RAM areas to the beginning of the user memory. Any blocks that you corrected before using COMPRESS MEMORY are still in the memory but are registered as invalid. The COMPRESS MEMORY function eliminates unused areas that resulted from deleting or correcting blocks. This function shifts a complete block to the beginning of the memory area. Ideally, one large available 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 smooth STOP modes. In the RUN mode, DBs and DXs that are longer than 512 data words are not shifted. In the STOP mode, all data 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, compression continues at the next unused memory area. Consequently, some unused areas can still remain after compression (see section 8.2).

#### 11.5 STATUS VARIABLES

You can call the STATUS VARIABLES programmer function to output the current signal status of specific operands (process variables).

This function activates system checkpoints in the RUN and smooth STOP modes. When program processing reaches the system checkpoint, the programmer displays the present signal status of the desired process variable. You can display the following process variables: inputs, outputs, flags, timers, counters. No addressing error (ADF) is triggered in the process image area.

Sequence of the STATUS VARIABLES Function in the RUN Mode:

If the STATUS VARIABLES function is running in the RUN mode, program processing continues until it reaches the system checkpoint. At the checkpoint, the system program scans the signal status of the operands and outputs this status to the programmer. The system program reads inputs from the **process image table**. As long as the function is not aborted, signal status is updated while program processing is running. The signal status is *not scanned at each* system checkpoint.

If program processing does not reach the system checkpoint, the system program does *not output* the signal status (e.g., in a continuous loop in the user program).

Sequence of the STATUS VARIABLES Function in the Smooth STOP Mode:

If the STATUS VARIABLES function is running in the smooth STOP mode, the programmer displays the signal status of the operands as the status exists at the system checkpoint. It is important to note that the system program scans the inputs **directly** from each peripheral module and outputs them. This permits testing to see if a peripheral input signal actually gets to the CPU. In multiprocessor operation you can display *all* inputs, not just the inputs indicated in DB 1.

### 11.6 STATUS

You can call the STATUS programmer function to test related operational sequences 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 programmer screen for every executed operation in the block (i.e., sequential operation). You can also use this function to test the parameter assignment of function blocks (i.e., block operation). The signal status of the actual operands is displayed.

#### Calling the STATUS Function and Specifying a Stopping Point

When you call the STATUS function on a programmer and enter the type and number of the block that you want to test (possibly including nesting sequence and search key), you are entering a stopping point.

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 stopping point (i.e. SEARCH:) in the correct nesting sequence (i.e. STATUS BLOCK:). Then the system program executes each of the monitored operations up to the operation boundary, outputting the processing results to the programmer.

Note: The results of operation processing are *not output in each of the subsequent scans*.

#### Nested Program Processing Levels and Interruptions

An operational sequence marked by a specified stopping point is completed even if a different program processing level (e.g., an error OB or interrupt OB) is activated and processed. Data that is altered when program processing levels change can be detected.

If an interruption in a nested program processing level puts the CPU into the STOP mode, data is output in the STOP mode. Data is output up to the operation that was executed before the program processing 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), smooth STOP (OB 39 only).

## 11.7 FORCE VARIABLES

You can call the FORCE VARIABLES programmer function to look at the values of operands (process variables) in the process image table and change them once. You can apply this function in the following modes: RUN, Smooth STOP.

You can also use this function within the PROGRAM TEST function. You can display the following process variables: flags, timers, counters, data words.

### Special Features:

The FORCE VARIABLES function has the following special features:

- No addressing error (ADF) is triggered in the process image area.
- Timeout (QVZ) signals are ignored in the peripherals area.
- In the area organized in bytes, the high byte is padded with FF (hexadecimal notation).
- Input byte "IB 0" (external process interrupts) is padded with 00xx (hexadecimal notation).

Any change becomes effective at a system checkpoint.

You can overwrite the controlled variables later (e.g., with your user program or process image update).

Note: The programmer controls the process variables byte by byte.

### NOTE

**If you are controlling several operands, the bytes are changed in memory in a sequence *distributed over several cycles*.**

## 11.8 FORCE

You can call the FORCE programmer function to manually set the output bytes of the programmable controller to the signal status you want. You can check the process devices connected to the outputs (e.g., motor, valve) and control them directly.

### NOTE

**The FORCE function is possible in the following situations: smooth STOP mode, hard STOP mode, within the PROGRAM TEST function.**

### Calling the FORCE Function

When you call the FORCE function in the STOP mode, the disable output command signal is suppressed (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 controlled byte by byte.

In multiprocessor operation, you can control all peripheral outputs (regardless of the peripheral assignment in DB 1).

Timeout errors that occur are detected when outputs are changed.

### Aborting the FORCE Function

You can abort the FORCE function by pressing the <BREAK> key on the programmer. 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 the call of the function and the actual control.

## 11.9 PROGRAM TEST

You can call the PROGRAM TEST programmer 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 programmer outputs the current signal status of operands, the accumulator contents, and the RLO for each operation executed.

### Calling the PROGRAM TEST Function and Specifying the First Stopping Point

You can call the PROGRAM TEST function in the RUN and smooth STOP modes. To call the PROGRAM TEST function, specify the type and number of the block you want to test. You may also want to include the nesting sequence. At the programmer, mark the first operation you want to test. This is how you specify the first stopping point.

When you specify the first stopping point during program processing, the CPU continues processing the program until it reaches the operation marked by the specified stopping point. 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 depth matches the block nesting depth that you specified. If the nesting depths do not match, the CPU continues program processing.

If program processing does not reach the specified stopping point (e.g., because the CPU goes into the STOP mode or there is a continuous loop in the user program), the programmer displays the message "Instruction will not be 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 programmer displays the data of the processed operation. The CPU waits for further instructions from the programmer.

### Calling the PROGRAM TEST Function in the Smooth STOP Mode:

You can also call the PROGRAM TEST function and specify an initial stopping point when the CPU is in the smooth STOP mode. The CPU remains in the smooth 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 proceeds as it does when you call the PROGRAM TEST function in the RUN mode.

### Executing the PROGRAM TEST Function and Specifying Another Stopping Point

Assuming that the CPU processed the operation at the first stopping point, you now have the following two options to continue the PROGRAM TEST function:

1. Specify the next operation as the following stopping point:

Move the cursor down to the next operation to specify the following stopping point. 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 programmer. However, if a nested program processing level interrupts operation processing at the following stopping point, the CPU processes the nested program first. Then the CPU returns to the following stopping point that you specified.

Important: You cannot specify a following stopping point when the CPU is in the STOP mode.

2. Specify a new stopping point:

At the programmer, specify any other operation in the same block or in a different block. The CPU continues program processing until it reaches the new stopping point. The CPU processes the operation up to the operation boundary, then it outputs the data.

### Retracting the Stopping Point

You can retract the stopping point by pressing the <BREAK> key if the CPU has not reached this specific stopping point. After that you can specify a new stopping point or call the END PROGRAM TEST function.

### Aborting the PROGRAM TEST Function

You can abort the PROGRAM TEST function during program processing and when the CPU is in the smooth STOP mode by calling the END PROGRAM TEST function. The CPU goes into the smooth STOP mode (or stays in that mode). The STOP LED flashes slowly. BEARB 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 programmer and the programmable controller is disconnected).



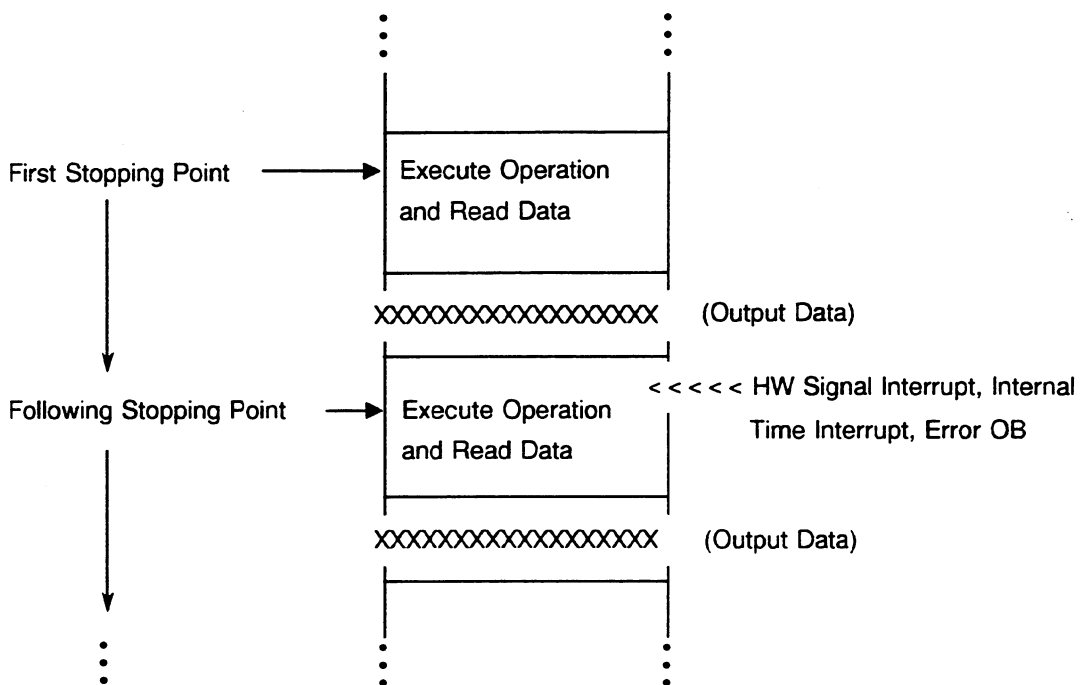
**Activating Other Program Processing Levels**

While the PROGRAM TEST function is running, the other program processing levels can be activated.

When an operation has been processed at a stopping point and a different program processing level is called at this location, the new program processing level is nested and completely processed before the *following stopping point* is processed. The new program processing level is nested and processed, if this program processing level can interrupt at operation boundaries.

**NOTE**

The system program reads data and outputs it at an operation boundary. At this point, all related program processing levels that have been activated have not been processed.



Note: If an operation has been processed at a stopping point and activation of a different program processing level is requested, you can set a stopping point at an operation in the different program processing level (e.g., you can look at a QVZ error OB directly after an operation that triggers a QVZ error).

**Interruptions during Program Processing**

- Program processing → STOP
- If one of the following stop conditions interrupts program processing before processing reaches the specified stopping point, the CPU goes into the STOP mode immediately:
- Multiprocessor stop
  - I/O not ready / PEU
  - RUN/STOP switch moved from RUN to STOP
  - Error OB not programmed etc.

If you execute a cold restart or a manual warm restart, the PROGRAM TEST function is still in effect and so is the stopping point is activated.

- Program processing at stopping point → STOP mode
- If one of the following stop conditions occurs at the stopping point or following stopping point during program processing, the CPU goes directly into the smooth STOP mode and outputs the data:
- RUN/STOP switch moved from RUN to STOP
  - STEP 5 operation STP
  - Error OB not programmed

If you do not specify a new stopping point while the CPU is in the smooth STOP mode, the PROGRAM TEST function is still in effect after a restart is performed.

#### NOTE

**If interruptions cause the CPU to go into the STOP mode during the PROGRAM TEST function, this function remains active after a restart is performed. The specified stopping point also remains active.**

While the PROGRAM TEST function is in effect, you can execute the following other functions at your programmer:

- OUTPUT ISTACK
- OUTPUT BSTACK
- LOAD BLOCK
- READ BLOCK
- DELETE BLOCK
- OUTPUT BLOCK LIST
- FORCE VARIABLES
- FORCE

### 11.10 OUTPUT ADDRESS

You can call the OUTPUT ADDRESS function to output the contents of memory and peripheral addresses in hexadecimal format at your programmer. You can reference all addresses for the following areas: RAM, EPROM/EEPROM, S5 bus, nonconfigured areas. No addressing error (ADF) is triggered in the process image area. No QVZ error occurs in the peripheral area.

In the area that you can address with bytes (i.e., flags, process image tables), the high byte is padded with FF.

### 11.11 MEMORY CONFIGURATION

You can plug two memory modules into the CPU 946/947. Each memory module can hold three different memory submodules. You can call the MEMORY CONFIGURATION function to transfer the following data about the CPU user memory to the programmer:

- allocation of memory module/submodule number
- memory submodule type (RAM/EPROM)
- start and end address of all detected memory submodules
- memory submodule capacity (16/32/64 Kwords)
- largest available user RAM memory block
- total of all available user RAM memory blocks

Summary of STEP 5 Operations for the CPU 946/947

Basic Operations

| Operation                   | Operand       |
|-----------------------------|---------------|
| • Boolean Logic Operations: |               |
| A I                         | 0.0 to 127.7  |
| A Q                         | 0.0 to 127.7  |
| A F                         | 0.0 to 255.7  |
| A D                         | 0.0 to 255.15 |
| A S                         | 0.0 to 4095.7 |
| A T                         | 0 to 255      |
| A C                         | 0 to 255      |
| AN I                        | 0.0 to 127.7  |
| AN Q                        | 0.0 to 127.7  |
| AN F                        | 0.0 to 255.7  |
| AN D                        | 0.0 to 255.15 |
| AN S                        | 0.0 to 4095.7 |
| AN T                        | 0 to 255      |
| AN C                        | 0 to 255      |
| O I                         | 0.0 to 127.7  |
| O Q                         | 0.0 to 127.7  |
| O F                         | 0.0 to 255.7  |
| O D                         | 0.0 to 255.15 |
| O S                         | 0.0 to 4095.7 |
| O T                         | 0 to 255      |
| O C                         | 0 to 255      |
| ON I                        | 0.0 to 127.7  |
| ON Q                        | 0.0 to 127.7  |
| ON F                        | 0.0 to 255.7  |
| ON D                        | 0.0 to 255.15 |
| ON S                        | 0.0 to 4095.7 |
| ON T                        | 0 to 255      |
| ON C                        | 0 to 255      |
| )                           |               |
| A(                          |               |
| O(                          |               |
| O                           |               |

| Operation                | Operand |
|--------------------------|---------|
| • Comparison Operations: |         |
| !=F                      |         |
| ><F                      |         |
| >F                       |         |
| >=F                      |         |
| <F                       |         |
| <=F                      |         |
| !=D                      |         |
| ><D                      |         |
| >D                       |         |
| >=D                      |         |
| <D                       |         |
| <=D                      |         |
| !=G                      |         |
| ><G                      |         |
| >G                       |         |
| >=G                      |         |
| <G                       |         |
| <=G                      |         |

## Summary of STEP 5 Operations

| Operation               |    | Operand                |
|-------------------------|----|------------------------|
| ● Set/Reset Operations: |    |                        |
| S                       | I  | 0.0 to 127.7           |
| S                       | Q  | 0.0 to 127.7           |
| S                       | F  | 0.0 to 255.7           |
| S                       | D  | 0.0 to 255.15          |
| S                       | S  | 0.0 to 4095.7          |
| R                       | I  | 0.0 to 127.7           |
| R                       | Q  | 0.0 to 127.7           |
| R                       | F  | 0.0 to 255.7           |
| R                       | D  | 0.0 to 255.15          |
| R                       | S  | 0.0 to 4095.7          |
| =                       | I  | 0.0 to 127.7           |
| =                       | Q  | 0.0 to 127.7           |
| =                       | F  | 0.0 to 255.7           |
| =                       | D  | 0.0 to 255.15          |
| =                       | S  | 0.0 to 4095.7          |
| ● Load Operations:      |    |                        |
| L                       | IB | 0 to 127               |
| L                       | IW | 0 to 126               |
| L                       | ID | 0 to 124               |
| L                       | QB | 0 to 127               |
| L                       | QW | 0 to 126               |
| L                       | QD | 0 to 124               |
| L                       | FY | 0 to 255               |
| L                       | FW | 0 to 254               |
| L                       | FD | 0 to 252               |
| L                       | DL | 0 to 255               |
| L                       | DR | 0 to 255               |
| L                       | DW | 0 to 255               |
| L                       | DD | 0 to 254               |
| L                       | T  | 0 to 255               |
| L                       | C  | 0 to 255               |
| L                       | PY | 0 to 127               |
| L                       | PW | 128 to 255             |
| L                       | OY | 0 to 126               |
| L                       | OW | 128 to 254             |
| L                       | SY | 0 to 255               |
| L                       | SW | 0 to 254               |
| L                       | SD | 0 to 4095              |
| LC                      | T  | 0 to 4094              |
| LC                      | C  | 0 to 4092              |
| L                       | KB | 0 to 255               |
| L                       | KS | 2 ASCII characters     |
| L                       | KM | bit pattern 16 bit     |
| L                       | KH | 0 to FFFF              |
| L                       | KF | -32768 to +32767       |
| L                       | KY | 0 to 255 each, 2 bytes |
| L                       | KT | 0.0 to 999.3           |
| L                       | KC | 0 to 999               |
| L                       | KG | 1)                     |
| L                       | DH | 0 to FFFF FFFF         |

| Operation                   |    | Operand    |
|-----------------------------|----|------------|
| ● Timer/Counter Operations: |    |            |
| SP                          | T  | 0 to 255   |
| SE                          | T  | 0 to 255   |
| SD                          | T  | 0 to 255   |
| SS                          | T  | 0 to 255   |
| SF                          | T  | 0 to 255   |
| R                           | T  | 0 to 255   |
| S                           | C  | 0 to 255   |
| R                           | C  | 0 to 255   |
| CU                          | C  | 0 to 255   |
| CD                          | C  | 0 to 255   |
| ● Transfer Operations:      |    |            |
| T                           | IB | 0 to 127   |
| T                           | IW | 0 to 126   |
| T                           | ID | 0 to 124   |
| T                           | QB | 0 to 127   |
| T                           | QW | 0 to 126   |
| T                           | QD | 0 to 124   |
| T                           | FY | 0 to 255   |
| T                           | FW | 0 to 254   |
| T                           | FD | 0 to 252   |
| T                           | DR | 0 to 255   |
| T                           | DL | 0 to 255   |
| T                           | DW | 0 to 255   |
| T                           | DD | 0 to 254   |
| T                           | PY | 0 to 127   |
| T                           | PW | 128 to 255 |
| T                           | OY | 0 to 126   |
| T                           | OW | 128 to 254 |
| T                           | SY | 0 to 255   |
| T                           | SW | 0 to 254   |
| T                           | SD | 0 to 4095  |

1)  $\pm 0.1469368 \times 10^{-38}$  to  
 $\pm 0.1701412 \times 10^{39}$

| Operation                | Operand   |
|--------------------------|-----------|
| ● Block Call Operations: |           |
| JC OB                    | 40 to 255 |
| JU OB                    | 40 to 255 |
| JU PB                    | 0 to 255  |
| JU FB                    | 0 to 255  |
| DOU FX                   | 0 to 255  |
| JU SB                    | 0 to 255  |
| JC PB                    | 0 to 255  |
| JC FB                    | 0 to 255  |
| DOC FX                   | 0 to 255  |
| JC SB                    | 0 to 255  |
| C DB                     | 2 to 255  |
| CX DX                    | 2 to 255  |
| G DB                     | 2 to 255  |
| GX DX                    | 2 to 255  |
| BE                       |           |
| BEC                      |           |
| BEU                      |           |
| ● Arithmetic Operations: |           |
| + F                      |           |
| - F                      |           |
| x F                      |           |
| : F                      |           |
| + G                      |           |
| - G                      |           |
| x G                      |           |
| : G                      |           |
| ● Other Operations:      |           |
| NOP 0                    |           |
| NOP 1                    |           |
| STP                      |           |
| BLD                      | 0 to 255  |
| BLD                      | 130       |
| BLD                      | 131       |
| BLD                      | 255       |

### Supplementary Operations

| Operation                       | Operand          |
|---------------------------------|------------------|
| ● Digital Operations:           |                  |
| AW                              |                  |
| OW                              |                  |
| XOW                             |                  |
| ● Timer/Counter Operations:     |                  |
| FR T                            | 0 to 255         |
| FR C                            | 0 to 255         |
| FR =                            | Formal operand   |
| SP =                            | Formal operand   |
| SD =                            | Formal operand   |
| SEC =                           | Formal operand   |
| SSU =                           | Formal operand   |
| SFD =                           | Formal operand   |
| ● Load and Transfer Operations: |                  |
| L =                             | Formal operand   |
| LD =                            | Formal operand   |
| LW =                            | Formal operand   |
| LWD =                           | Formal operand   |
| T =                             | Formal operand   |
| L RI                            | 0 to 255         |
| L RJ                            | 0 to 255         |
| L RS                            | 0 to 255         |
| L RT                            | 0 to 255         |
| T RI                            | 0 to 255         |
| T RJ                            | 0 to 255         |
| T RS                            | 0 to 255         |
| T RT                            | 0 to 255         |
| LY GB                           | -32768 to +32767 |
| LY GW                           | -32768 to +32767 |
| LY GD                           | -32768 to +32767 |
| TY GB                           | -32768 to +32767 |
| TY GW                           | -32768 to +32767 |
| TY GD                           | -32768 to +32767 |
| LW GW                           | -32768 to +32767 |
| LW GD                           | -32768 to +32767 |
| TW GW                           | -32768 to +32767 |
| TW GD                           | -32768 to +32767 |
| LY CB                           | -2048 to +2047   |
| LY CW                           | -2048 to +2047   |
| LY CD                           | -2048 to +2047   |
| TY CB                           | -2048 to +2047   |

## Summary of STEP 5 Operations

| Operation                               | Operand          |
|---|------------------|
| ● Load and Transfer Operations (cont.): |                  |
| TY CW                                   | -2048 to +2047   |
| TY CD                                   | -2048 to +2047   |
| LW CW                                   | -2048 to +2047   |
| LW CD                                   | -2048 to +2047   |
| TW CW                                   | -2048 to +2047   |
| TW CD                                   | -2048 to +2047   |
| TSC                                     | -2048 to +2047   |
| TSG                                     | -32768 to +32767 |
| ● Binary Logic Operations:              |                  |
| A =                                     | Formal operand   |
| AN =                                    | Formal operand   |
| O =                                     | Formal operand   |
| ON =                                    | Formal operand   |
| ● Conversion Operations:                |                  |
| CFW                                     |                  |
| CSW                                     |                  |
| CSD                                     |                  |
| DEF                                     |                  |
| DUF                                     |                  |
| DED                                     |                  |
| DUD                                     |                  |
| FDG                                     |                  |
| GFD                                     |                  |
| ● Shift and Rotate Operations:          |                  |
| SLW                                     | 0 to 15          |
| SRW                                     | 0 to 15          |
| SLD                                     | 0 to 32          |
| SSD                                     | 0 to 32          |
| RLD                                     | 0 to 32          |
| RRD                                     | 0 to 32          |
| SSW                                     | 0 to 15          |

| Operation               | Operand          |
|-------------------------|------------------|
| ● Jump Operations:      |                  |
| JU =                    | Symbolic address |
| JC =                    | Symbolic address |
| JZ =                    | Symbolic address |
| JN =                    | Symbolic address |
| JP =                    | Symbolic address |
| JM =                    | Symbolic address |
| JO =                    | Symbolic address |
| JOS =                   | Symbolic address |
| ● Set/Reset Operations: |                  |
| S =                     | Formal operand   |
| RB =                    | Formal operand   |
| RD =                    | Formal operand   |
| = =                     | Formal operand   |
| SU I                    | 0.0 to 127.7     |
| SU Q                    | 0.0 to 127.7     |
| SU F                    | 0.0 to 255.7     |
| SU T                    | 0 to 255.15      |
| SU C                    | 0 to 255.15      |
| SU D                    | 0.0 to 255.15    |
| SU RI                   | 0.0 to 255.15    |
| SU RJ                   | 0.0 to 255.15    |
| RU I                    | 0.0 to 127.7     |
| RU Q                    | 0.0 to 127.7     |
| RU F                    | 0.0 to 255.7     |
| RU T                    | 0 to 255.15      |
| RU C                    | 0 to 255.15      |
| RU D                    | 0.0 to 255.15    |
| RU RI                   | 0.0 to 255.15    |
| RU RJ                   | 0.0 to 255.15    |
| ● Other Operations:     |                  |
| ACR                     |                  |
| RA                      |                  |
| IA                      |                  |
| IAE                     |                  |
| RAE                     |                  |
| BAS                     |                  |
| BAF                     |                  |
| ENT                     |                  |
| D                       | 0 to 255         |
| I                       | 0 to 255         |

| Operation                   | Operand        |
|-----------------------------|----------------|
| ● Other Operations (cont.): |                |
| SED                         | 0 to 31        |
| SEE                         | 0 to 31        |
| DO =                        | Formal operand |
| DO DW                       | 0 to 255       |
| DO FW                       | 0 to 255       |
| ● Bit Test Operations:      |                |
| TB I                        | 0.0 to 127.7   |
| TB Q                        | 0.0 to 127.7   |
| TB F                        | 0.0 to 255.7   |
| TB T                        | 0 to 255.15    |
| TB C                        | 0 to 255.15    |
| TB D                        | 0.0 to 255.15  |
| TB RI                       | 0.0 to 255.15  |
| TB RJ                       | 0.0 to 255.15  |
| TB RS                       | 0.0 to 255.15  |
| TB RT                       | 0.0 to 255.15  |
| TBN I                       | 0.0 to 127.7   |
| TBN Q                       | 0.0 to 127.7   |
| TBN F                       | 0.0 to 255.7   |
| TBN T                       | 0 to 255.15    |
| TBN C                       | 0 to 255.15    |
| TBN D                       | 0.0 to 255.15  |
| TBN RS                      | 0.0 to 255.15  |
| TBN RJ                      | 0.0 to 255.15  |
| TBN RS                      | 0.0 to 255.15  |
| TBN RT                      | 0.0 to 255.15  |

**System Operations**

| Operation                    | Parameter          |
|------------------------------|--------------------|
| ● Load /Transfer Operations: |                    |
| LIR                          | 0 to 14            |
| TIR                          | 0 to 14            |
| LDI                          | Reg. name          |
| TDI                          | Reg. name          |
| TNW                          | 0 to 255           |
| TXB                          |                    |
| TXW                          |                    |
| MBR                          | Constant (20 bits) |

| Operation                           | Operand            |
|-------------------------------------|--------------------|
| ● Load/Transfer Operations (cont.): |                    |
| ABR                                 | - 32768 to + 32767 |
| LRW                                 | "                  |
| LRD                                 | "                  |
| TRW                                 | "                  |
| TRD                                 | "                  |
| MAS                                 |                    |
| MAB                                 |                    |
| MSA                                 |                    |
| MSB                                 |                    |
| MBA                                 |                    |
| MBS                                 |                    |
| ● Set/Reset Operations:             |                    |
| SU RS                               | 60.0 to 63.15      |
| SU RT                               | 0.0 to 255.15      |
| RU RS                               | 60.0 to 63.15      |
| RU RT                               | 0.0 to 255.15      |
| ● Block Call Operations:            |                    |
| JU OB                               | 1 to 39            |
| JC OB                               | 1 to 39            |
| JUR                                 | -32768 to +32767   |
| ● Arithmetic Operations:            |                    |
| ADD BN                              | -128 to +127       |
| ADD KF                              | -32768 to +32767   |
| ADD DH                              | 0 to FFFF FFFF     |
| + D                                 |                    |
| - D                                 |                    |
| ● Other Operations:                 |                    |
| DI                                  |                    |
| DO RS                               | 0 to 255           |
| TAK                                 |                    |
| STS                                 |                    |
| STW                                 |                    |
| SIM                                 |                    |
| LIM                                 |                    |

**A**

Absolute address, 1-7, 1-9, 9-1 to 9-20  
 Accumulator 1 (ACCU 1), 2-4, 3-11, 3-12, 3-17, 3-18, 3-20 to 3-22  
 Accumulator 2 (ACCU 2), 2-4, 3-11, 3-12, 3-17, 3-18  
 Accumulators (ACCUs), 1-7, 2-4, 3-11, 3-52, 3-54, 9-9  
 ACCUs (See Accumulators)  
 Actual operands, 2-18, 2-20, 2-22, 2-25, 2-26, 3-46, 3-50  
 Addressable system data area, 8-35 to 8-37  
 Address counter, STEP, 1-7, 9-14  
 Addressing error, 3-72, 5-2, 5-11, 5-15  
 AND-before-OR operation, 3-28  
 AND operation, 3-13, 3-15, 3-27  
 Arithmetic  
   double-word, 1-8  
   floating-point, 1-8  
 Arithmetic operations, 2-1, 2-5, 3-11, 3-13, 3-24, 3-70

**B**

Base address register, 1-7, 9-10, 9-12 to 9-13  
 Basic clock rate, 1-15, 1-16  
 Basic operations, 2-1, 3-14 to 3-45, 12-1 to 12-3  
 "BASP" (See Disable output command signal)  
 BCD code, numbers in, 2-5, 2-8  
 Binary operations, 2-4  
 Bit condition codes, 3-12 to 3-13  
 Bit test operations, 3-47  
 Block body, 2-10  
 Block call, 2-13 to 2-14, 2-25  
 Block call operations, 1-11, 3-25, 3-70  
 Block end operation BE, 2-10, 2-13, 2-14, 2-31, 3-25  
 Blocks, 2-3, 2-9 to 2-35  
   address, 5-3  
   address list, 2-35, 3-4, 8-11  
   body, 2-10  
   boundaries, 1-11, 1-15, 4-4  
   generation, 6-12, 6-14  
   header, 2-10, 8-10  
   length, 2-10  
   location in user memory, 8-10  
   nesting, 1-15, 3-1, 3-4, 3-5  
   number, 2-9, 2-10, 5-3  
   preheader, 2-11  
   programming, 2-13  
   types, 1-5, 2-3, 2-9, 2-10, 2-11  
 Block stack (BSTACK), 5-3 to 5-4  
   overflow, 3-72, 4-9  
 Block start address (BA), 3-4  
 Boolean logic operations, 3-14 to 3-15, 3-27 to 3-30

supplementary, 3-46  
 Bracket counter overflow (KZU), 3-71, 3-72, 4-9  
 Bracket depth, 1-15  
 BSTACK (See Block stack)  
 Bus-enable signal, 10-2, 10-6

**C**

CC 1 and CC 0 (See Condition codes)  
 Central processing units (CPUs), 1-1 to 1-18  
 Checkpoints, system, 11-1  
 Clock, real-time, 1-8, 1-11, 8-30 to 8-34  
 Closed-loop control, 1-1, 1-8  
 Cold restart, 2-17, 2-35, 4-12, 4-13, 4-15 to 4-18  
   retentive, 4-16, 4-17, 7-4, 7-5  
 Communications processors (CPs), 1-2, 1-8, 3-9, 10-1, 10-3, 10-5, 10-7  
 Comparison operations, 2-4, 2-5, 3-11, 3-13, 3-16 to 3-17, 3-44 to 3-45  
 COMPRESS MEMORY programmer function, 1-12, 2-12, 11-3  
 Condition codes, 3-12 to 3-13, 3-18, 3-52, 3-53, 3-71, 5-5  
 Condition code mask, 3-71  
 Control bits, 5-5 to 5-7  
 Controller mode, 1-15 to 1-17  
   150U, 1-9, 1-11, 1-15, 1-16, 4-4, 4-22, 4-23, 7-4  
   155U, 1-4, 1-15 to 1-17, 4-4, 4-23, 4-31, 7-4  
 Control system flowchart (CSF) method of representation, 2-2  
 Conversion operations, 2-5, 3-56 to 3-58  
 Coordinator 923C, 1-1, 1-16, 1-17, 10-2, 10-4, 10-7, 10-12  
 Counter, 1-6, 1-7, 1-9, 2-21, 2-28, 4-21  
 Counter operations, 3-20, 3-40 to 3-43  
   supplementary, 3-48 to 3-49  
 CPU 946/947, 1-1 to 1-4, 1-6 to 1-9, 1-16  
   application, 1-8  
   new features and functions, 1-11, 1-13 to 1-14  
 CSF (See Control system flowchart method of representation)  
 Cycle time, 1-4, 1-5, 2-35, 3-8  
   setting in extended data block DX 0, 7-5  
 Cycle time exceeded error (ZYK), 3-72, 5-2, 5-11, 5-17  
 Cycle time monitoring, 1-4, 3-7, 4-22  
 Cyclic program processing, 1-4, 3-1, 3-7 to 3-10, 4-3, 4-21 to 4-23



**D**

Data block DB 0, 2-11, 2-35, 8-11  
Data block DB 1, 2-35, 10-2, 10-8 to 10-10  
Data blocks, 1-5, 1-6, 2-3, 2-10, 2-11, 2-30 to 2-35  
    copy, 6-21  
    current, 1-7  
    length (DBL), 3-4  
    opening, 2-32 to 2-34  
    programming, 2-31  
    structure, 2-30 to 2-31  
    transfer, 2-30, 2-32  
    validity area, 2-34  
Data exchange  
    between CPUs, 10-3 to 10-4  
    between CPU(s) and CP(s), 10-5  
Data format, 2-5 to 2-7, 2-11, 2-22, 2-30, 2-31  
DB0 (See Data block DB 0)  
DB1 (See Data block DB 1)  
DBL (See Data block length)  
Decimal numbers, 2-5, 2-8  
Decrement operation, 3-60  
DELETE programmer function, 11-2  
Delete STEP 5 blocks, 6-9 to 6-11  
Digital functions, 3-11  
Digital logic operations, 3-13  
Digital operations, 3-52  
DIRECTORY programmer function, 11-2  
Disable output command signal ("BASP"), 4-7, 4-12, 4-21  
Disable semaphore, 3-64 to 3-69  
Double-word arithmetic, 1-8  
Dual-port RAM  
    accessing, 9-17 to 9-20  
    pages, 5-9, 8-1, 8-5, 9-17 to 9-20  
Duplicating a data block, 6-21  
DXs (See Extended data blocks)  
DX0 (See Extended data block DX 0)

**E**

Enable semaphore, 3-64 to 3-69  
EPROM submodule, 1-6, 2-3, 2-11, 3-6  
ERAB (See First bit scan)  
Error, 1-5, 1-7, 1-16  
    addressing, 5-15  
    causes, 5-14 to 5-19  
    cycle time exceeded, 5-17  
    diagnostics, 5-1 to 5-19  
    load/transfer, 2-33, 5-15  
    parity, 3-72, 5-17  
    substitution, 5-14  
    timeout, 5-16 to 5-17  
    user, 8-21 to 8-23  
Error handling, organization blocks, 4-4, 5-12 to 5-13, 5-14 to 5-18  
Error IDs, 5-6, 5-8  
Error numbers

data block DB 1, 8-23  
extended data block DX 0, 8-23  
in RS 75, 8-21

Error organization blocks, 1-4  
Expansion unit (EU), 1-2  
Extended data block DX 0, 1-11 to 1-13, 1-15, 1-16, 2-11, 2-35, 7-1 to 7-9, 10-2  
    assigning parameters, 7-7 to 7-9  
    programming, 3-10  
    structure, 7-2 to 7-3  
Extended data block DX 1, 2-11, 2-35  
Extended data blocks (DXs), 1-8, 1-11, 1-13, 2-9, 2-10, 2-11, 2-30 to 2-35  
Extended function blocks (FXs), 1-8, 1-11, 2-9, 2-10, 2-18 to 2-29  
Extended interface data area (RJ), 3-51, 8-12  
Extended system data area (RT), 3-51, 8-13  
External process interrupts, 1-15, 3-59, 4-3, 4-21, 4-23, 4-28 to 4-29

**F**

Filler blocks, 1-10, 8-8 to 8-9  
First bit scan (ERAB), 3-12, 3-15  
Fixed-point number, 2-5, 2-6, 2-23  
Flag,  
    interprocessor communication, 1-5, 1-6, 1-13, 1-17, 3-7, 3-9, 10-3 to 10-13  
Floating-point number, 2-5 to 2-7, 2-23  
FORCE programmer function, 11-5  
FORCE VARIABLES programmer function, 11-5  
Formal operand, 2-21, 3-50  
Function blocks (FBs), 2-3, 2-10, 2-18 to 2-29  
    assigning parameters, 2-25 to 2-27  
    calling, 2-25 to 2-27  
    extended (FXs), 1-8, 1-11, 2-9, 2-10, 2-18 to 2-29  
    programming, 2-21 to 2-24  
    structure, 2-19 to 2-20

**G**

Gaps, memory, 1-10, 2-12, 8-8  
Generate STEP 5 blocks, 6-12 to 6-14  
Generating the RLO, 3-15, 3-18  
Global memory, 3-64, 3-65, 9-1 to 9-4, 9-14 to 9-17  
GRAPH 5 method of representation, 2-2

**H**

"Halt" signal, 4-14  
Handling blocks, 10-7  
Hard STOP mode, 4-1, 4-2, 4-11  
Hot restart, 4-19

- I**
- Increment operation, 3-60
  - Input byte "IB 0," 1-15, 2-16, 4-28 to 4-29
  - Input/output modules (I/Os), 1-2, 1-3, 3-19
  - Integrated special functions, 6-1 to 6-24
  - Intelligent input/output modules (IPs), 1-2
  - Interface data area (RI), 3-51, 8-12
  - Interprocessor communication flags, 1-5, 1-6, 1-13, 1-17, 3-9, 4-21, 4-22, 7-4, 10-3 to 10-13
    - inputs/outputs, 2-35, 10-3 to 10-5
  - Interrupt driven program processing, 1-4, 1-5, 3-8, 3-9, 4-3, 4-20, 4-21, 4-27 to 4-33
  - Interrupt priority, 1-12
  - Interrupts, 1-11
  - Interrupt points, 3-9
  - Interrupt signal, 2-16
  - Interrupt stack (ISTACK), 5-1 to 5-19
    - overflow (STUEU), 3-72, 5-11, 5-14
  - I/Os (See Input/output modules)
  - IPC flags (See Interprocessor communication flags)
  - IPs (See Intelligent input/output modules)
- J**
- Jump operations, 3-11, 3-52 to 3-53, 3-70
- L**
- LAD (See Ladder diagram method of representation)
  - Ladder diagram (LAD) method of representation, 2-2
  - Length register, data block (DBL), 9-8
  - Library number, 2-10
  - Load/transfer error, 1-10, 2-33, 5-15
  - Load/transfer operations
    - basic, 3-16 to 3-17
- M**
- Memory
    - access using absolute addresses, 9-1 to 9-20
    - gaps, 1-10, 2-12, 8-8
    - global, 3-64, 3-65
    - system, 2-11
  - Memory areas, 1-6, 1-7, 1-13, 8-1 to 8-8
  - Memory assignment, 8-1 to 8-37
    - CPU 946/947, 8-2 to 8-7
    - peripherals, 8-5 to 8-6
    - system RAM, 8-3 to 8-4
  - Memory blocks, transferring, 9-10
  - Memory module, 355, 1-16, 2-3, 2-11, 3-6, 10-1
  - Memory organization, 8-1 to 8-37, 9-1 to 9-4
    - CPU 946/947, 8-8 to 8-37
  - Memory register, addressing, 1-7
  - MEMORY CONF programmer function, 11-9
- Methods of representation, 2-2
  - control system flowchart (CSF), 2-1, 2-2
  - GRAPH 5, 2-2
  - ladder diagram (LAD), 2-1, 2-2
  - statement list (STL), 2-1, 2-2
- Modes of operation, 4-1 to 4-33
  - hard STOP, 4-1, 4-2, 4-11
  - RESTART, 4-1, 4-2, 4-3, 4-12 to 4-20
  - RUN, 4-1, 4-2, 4-3, 4-21 to 4-33
  - smooth STOP, 4-1, 4-2, 4-3, 4-5 to 4-10
  - STOP, 4-6 to 4-11
- Monitoring, 1-8
- Multiprocessing, 1-3, 1-8, 1-13, 1-15 to 1-17, 10-1 to 10-13
- Multiprocessor communication, 6-19, 10-7
- Multiprocessor operation, 1-2, 1-13, 1-16, 7-9
- N**
- NAU (See Power failure)
  - Nesting depth, block, 3-4, 3-5
  - "No" operations, 3-26
  - Numbers
    - BCD code, 2-5, 2-8
    - fixed-point, 2-5, 2-6, 2-23
    - floating-point, 2-5, 2-6, 2-7, 2-23
- O**
- "O" peripherals, 3-19
  - Operand, 2-4, 2-10
    - actual, 2-18, 2-20, 2-22, 2-25, 2-26
    - formal, 2-18 to 2-21, 2-24, 2-25, 3-46, 3-50
  - Operating status, 5-2, 5-5
  - Operation
    - boundaries, 1-11, 4-4
  - OR-before-AND operation, 3-29
  - OR operation, 3-27
  - Or condition code (OR), 3-13
  - Organization block 1 (OB 1), 3-1, 3-2, 4-22
  - Organization blocks (OBs), 2-3, 2-9, 2-13 to 2-17, 3-1, 5-12
    - programming, 3-10
    - special function, 6-1
  - Other operations, 3-59
  - OUTPUT ADDRESS programmer function, 11-9
  - Overall reset, 1-8, 3-6, 4-9 to 4-10
  - Overflow, 3-53, 3-72
    - condition code (OV), 3-13
    - interrupt stack (STUEU), 5-11, 5-14
    - stored (OS), 3-13, 3-18, 3-53
- P**
- Paragraph address, 1-9, 1-10, 8-8, 8-9, 8-11
  - Parity error, 3-72, 5-17
  - Peripheral address
    - area, 1-12
  - Peripheral area, 1-6, 1-7, 1-9

- Peripheral modules, 1-1 to 1-3, 1-6, 3-8
  - Peripherals
    - accessing, 8-7
    - memory assignment, 8-5
  - PII (See Process image input table)
  - PIQ (See Process image output table)
  - Power failure (NAU), 3-72, 4-13, 4-14, 4-20
  - "P" peripherals, 3-8, 3-19
  - Priorities, 4-25, 4-27, 4-29, 4-31
  - PROGRAM TEST programmer function, 11-6 to 11-9
  - Process image input (PII) table, 1-4, 1-6, 1-7, 3-8, 3-19, 4-18, 4-22
  - Process image output (PIQ) table, 1-4 to 1-7, 3-8, 3-19, 4-22
  - Processing
    - interrupt driven, 1-4
    - time driven, 1-4, 1-15, 1-16
  - Program blocks (PBs), 2-3, 2-9, 2-10, 2-13, 2-18
    - calling, 2-13
  - Program memory, 2-11
  - Programmer function
    - BSTACK, 5-3 to 5-4
    - COMPRESS MEMORY, 1-12, 2-12, 6-9, 6-13, 6-21, 6-22, 8-8, 11-3
    - DELETE, 11-2
    - DELETE PC, 4-10
    - DIRECTORY, 11-2
    - END PROGRAM TEST, 4-9, 4-13
    - FORCE, 11-5
    - FORCE VARIABLES, 11-5
    - ISTACK, 5-2
    - MEMORY CONF, 11-9
    - OUTPUT ADDRESS, 11-9
    - PC START, 4-15
    - PC STOP, 4-8, 4-14
    - PROGRAM TEST, 11-6 to 11-9
    - START, 4-13, 11-2
    - STATUS, 3-12, 11-4
    - STATUS VARIABLES, 11-3
    - STOP, 11-2
  - Programming examples
    - copying a data block from EPROM to RAM, 6-21
    - disabling interrupts, 6-11
    - duplicating a data block in RAM, 6-22
    - generating STEP 5 blocks, 6-12
    - logic, set/reset, timer, counter, and comparison operations, 3-27 to 3-45
    - reading the time of day, 6-6 to 6-7
    - setting the time of day, 6-6 to 6-7
    - supplementary timer and counter operations, 3-48
    - transferring process image tables, 6-15 to 6-18
  - Program processing, 2-4, 2-9
    - cyclic, 1-4, 3-1, 3-7 to 3-10, 4-3, 4-7, 4-18 to 4-20, 4-22 to 4-24, 7-1, 7-4
    - hardware (HW) signal interrupt driven, 1-4
    - interrupt driven, 1-4, 1-5, 3-8, 3-9, 4-3, 4-20, 4-27 to 4-33, 7-1, 7-5
    - time driven, 1-4, 1-5, 1-15, 1-16, 3-8, 3-9, 4-3, 4-20, 4-23 to 4-33, 7-1, 7-4
  - Program processing level, 3-1, 3-8, 4-1 to 4-5
- R**
- RAM, 1-15
    - area, 1-9
    - submodules, 1-6, 2-3, 2-12, 2-11, 3-6
  - Reading the time of day, 6-4 to 6-7
  - Real-time clock, 1-8, 1-11
  - Register, 2-4, 3-4
    - base address, 1-7, 9-10, 9-12 to 9-13
    - block start address, 9-7
    - CPU 946/947, 1-6, 1-7
    - data block length, 1-7, 9-8
    - data block start address, 1-7, 9-7
    - system data, 8-15 to 8-37
  - Restart, hot, 4-19
  - RESTART mode, 4-1, 4-2, 4-3, 4-12 to 4-19, 4-21
    - interrupts, 4-19
  - Restart types
    - comparing in multiprocessor operation, 6-20
  - Result of logic operation (RLO), 2-4, 2-14, 3-13, 3-15
  - Retentive cold restart, 4-16, 7-4, 7-5
  - Return address, 3-4, 5-3, 5-19
  - RI (See Interface data area)
  - RJ (See Extended interface data area)
  - RLO (See Result of logic operation)
  - Rotate operations, 3-54 to 3-55
  - RS (See System data area)
  - RS register, 1-11
  - RT area (See Extended system data area)
  - RUN mode, 2-12, 4-1, 4-2, 4-3, 4-21 to 4-33
  - RUN/STOP switch, 3-72, 4-6, 4-8, 4-9, 4-10, 4-12 to 4-14, 4-20
- S**
- S flags, 1-6, 1-14
  - S5 bus, 1-2, 1-3, 1-6, 1-8, 1-12, 1-15, 1-16, 2-16, 4-30, 4-31
  - Semaphores, 1-11, 3-64 to 3-69
    - application example, 3-66 to 3-69
    - disable, 3-64 to 3-69
    - enable, 3-64 to 3-69
  - Sequence blocks (SBs), 2-3, 2-10, 2-13, 2-14
  - Set operations in the RS/RT area, 9-20
  - Set/reset operations, 3-32 to 3-34
    - basic, 3-15

- programming examples, 3-31 to 3-34
  - supplementary, 3-47
  - Setting a counter, 3-40
  - Setting the time of day, 6-4 to 6-7
  - Shift operations, 3-13, 3-54 to 3-55
  - Signal status
    - negating, 3-15
    - scanning, 3-15
  - Smooth STOP mode, 1-8, 1-11, 1-13, 2-16, 3-26, 4-1, 4-2, 4-3, 4-6 to 4-10, 11-7
  - Special functions
    - integrated, 6-1 to 6-24
  - STA (See Status condition code)
  - Start address register, data block, 1-7
  - Start IDs, system data register RS8, 8-19
  - START programmer function, 11-2
  - Start-up procedure, 1-5, 1-13, 10-11
  - Statement list (STL) method of representation, 2-2, 3-46,
  - Status condition code (STA), 3-13
  - STATUS programmer function, 3-12, 11-4
  - STATUS VARIABLES programmer function, 11-3
  - STEP 5 operations, 1-7, 1-12, 1-13, 2-4, 2-10, 3-11
  - STEP 5 programming language, 2-1
  - STEP address counter (SAC), 1-7, 9-10
  - STEP address register, 9-10, 9-14
  - STL (See Statement list method of representation)
  - STOP mode, 1-4, 1-12, 1-16, 2-12, 4-6 to 4-11
    - hard, 4-1, 4-2, 4-11
    - smooth, 1-8, 2-16, 4-1, 4-2, 4-3, 4-6 to 4-10
  - Stop operation, 3-26
  - STOP programmer function, 11-2
  - Stopping point, 11-9
  - Stored overflow (OS), 3-13, 3-53
  - Structure
    - system, 1-1 to 1-2
  - Substitution error, 2-29, 3-72, 5-14
  - Supplementary operations, 2-1, 2-18, 3-11, 3-46 to 3-69, 12-3 to 12-5
  - Symbolic representation, 2-4
  - System checkpoints, 11-1
  - System data area (RS), 1-9, 3-51
  - System data registers, 8-15 to 8-37
  - System interrupt, 3-72
  - System memory area (RS, RT), 8-1, 8-13 to 8-37
  - System operations, 2-1, 3-11, 3-70 to 3-72
  - System program, 1-5, 1-6, 1-10, 1-15, 1-16, 2-3, 2-9, 3-1, 3-4, 3-6 to 3-9
    - 1-15, 1-16, 3-8, 3-9, 4-3, 4-20, 4-23 to 4-33
  - Time interrupts, 4-3, 4-23, 4-25 to 4-33
  - Timeout error, 3-72, 5-16 to 5-17
  - Timer, 1-6, 1-7, 2-21, 2-28, 4-7, 4-12, 4-18, 4-21, 4-24
  - Timer operations, 3-35 to 3-39
    - basic, 3-20
    - programming examples, 3-35 to 3-39
    - supplementary, 3-48 to 3-49
  - Transfer operations, 3-16 to 3-18, 3-50, 3-51
  - Transfer error (See Load/transfer error)
- U**
- User interface, organization block OB 1, 4-22
  - User memory, 1-6, 1-9, 1-11, 1-15, 3-4, 3-6
  - User program, 1-4, 1-5 to 1-8, 1-16, 2-1 to 2-35, 3-1, 3-2, 3-3, 3-6 to 3-10, 5-1
- W**
- Warm restart, 2-17, 2-35, 4-3, 4-12 to 4-16, 4-18 to 4-21
  - Word condition codes, 3-13
- T**
- Test operation, 10-12
  - Time driven program processing, 1-4, 1-5,

# SIEMENS

## SIMATIC S5

Appendix  
for Manual S5-155U (CPU 946/947)

---

Notes

C79000-A8576-C647-01

---

## Ordering Information

In this section you will find the order numbers of the products mentioned in the manual. The order numbers are listed according to the parts in which the products are mentioned. Please also refer to the current catalogs.

| <b>For Part 2</b>   | <b>Order Number</b> |
|---|---------------------|
| <b>Power supply unit with fan for central controller</b>                |                     |
| 230/120 V AC, floating, 5 V, 40 A                                       | 6ES5 955-3LF12      |
| 24 V DC, floating, 5 V, 40 A  | 6ES5 955-3NF11      |
| 15-V supplementary module   | 6ES5 956-0AA12      |
| <b>Power supply unit with fan for S5-183 and S5-185 expansion units</b> |                     |
| 230/120 V AC, floating, 5 V, 18 A                                       | 6ES5 955-3LC14      |
| 230/120 V AC, floating, 5 V, 40 A                                       | 6ES5 955-3LF12      |
| 24 V DC, non-floating, 5 V, 10 A  | 6ES5 955-3NA12      |
| 24 V DC, floating, 5 V, 18 A  | 6ES5 955-3NC13      |
| 24 V DC, floating, 5 V, 40 A  | 6ES5 955-3NF11      |
| <b>Fan unit (only for S5-184)</b>                                       |                     |
| 240/120 V AC  | 6ES5 988-3LA11      |
| 24 V DC   | 6ES5 988-3NA11      |
| <b>Load supply 951</b>  | 6ES5 951-4LB11      |
| <b>Enable supply 958</b>  | 6ES5 958-4UA11      |
| <b>Accessories</b>  |                     |
| Air baffle  | 6ES5 981-0DA11      |
| Dust filter holder  | 6ES5 981-0FA11      |
| Dust filter (10 off)  | 6ES5 981-0EA11      |
| Replacement fan   |                     |
| for 6ES5 955-3LC14  |                     |
| 6ES5 955-3LF12  |                     |
| 6ES5 988-3LA11  | 6ES5 998-3LB21      |
| for 6ES5 955-3NC13  |                     |
| 6ES5 955-3NA12  |                     |
| 6ES5 955-3NF11  |                     |
| 6ES5 988-3NA11  | 6ES5 988-3NB11      |

**Order Number**

|                     |                |
|---------------------|----------------|
| Fuses (6.3 x 32 mm) |                |
| 15 A, slow-blow     | 299461         |
| 6 A, quick-blow     | 300095         |
| 4 A, quick-blow     | 291963         |
| Dummy front panels  |                |
| 1 slot wide         | 6XF2 008-6KB00 |
| 2 slots wide        | 6XF2 016-6KB00 |

**For Part 3**

**Central controller**

|                        |                |
|------------------------|----------------|
| with power supply unit |                |
| 6ES5 955-3LF12         | 6ES5 155-3UA11 |
| 6ES5 955-3NF11         | 6ES5 155-3UA21 |

**Accessories**

|   |                |
|---|----------------|
| Back-up battery for power supply plug-ins | 6EW1 000-7AA   |
| Air baffle                                | 6ES5 981-0DA11 |

**For Parts 4 and 5**

|   |                |
|---|----------------|
| Memory module 355                                 |                |
| with 3 slots for 373 and 377 memory module (long) | 6ES5 355-3UA11 |

**EPROM submodule 373**

|                             |                |
|-----------------------------|----------------|
| 32 x 2 <sup>10</sup> bytes  | 6ES5 373-0AA41 |
| 64 x 2 <sup>10</sup> bytes  | 6ES5 373-0AA61 |
| 128 x 2 <sup>10</sup> bytes | 6ES5 373-0AA81 |

**RAM submodule 377**

|                             |                |
|-----------------------------|----------------|
| 32 x 2 <sup>10</sup> bytes  | 6ES5 377-0AB21 |
| 64 x 2 <sup>10</sup> bytes  | 6ES5 377-0AB31 |
| 128 x 2 <sup>10</sup> bytes | 6ES5 377-0AB41 |

**Spare parts**

|                                       |                |
|---------------------------------------|----------------|
| Back-up battery for RAM submodule 377 | 6ES5 980-0DA11 |
|---------------------------------------|----------------|

**Order Number**

**For Part 6**

923C coordinator

6ES5 923-3UC11

**Spare parts for 923C coordinator**

Coding plugs

C79334-A3011-B12

Front cover

C79451-A3079-C251

**Connecting cable 725**

from 923C coordinator to CP 530, 143 and 5430

0.9 m

6ES5 725-0AK00

2.5 m

6ES5 725-0BC50