SIEMENS

SIMATIC S5

Handling Blocks for the S Processor S5-135U Programmable Controller

Description

Order No. C79000-G8576-C325-02

Contents

Warning	C79000-R8576-C325
Information	
Suggestions/Corrections	

Description C79000-B8576-C325-03

SIEMENS

Warnhinweis

Gefahren beim Einsatz sogenannter SIMATIC-kompatibler Baugruppen fremder Hersteller

"Den Hersteller eines Produktes (hier SIMATIC) trifft die Produktbeobachtungspflicht, d. h. er muß generell vor Gefahren des Produktes warnen. Diese Produktbeobachtungspflicht wurde von der neueren Rechtssprechung auch auf fremde Zubehörteile erstreckt. Der Hersteller hat danach die Verpflichtung, auch solche Gefahren zu beobachten und zu erkennen, die aus der Verbindung des Produktes mit Produkten anderer Hersteller entstehen.

Aus diesem Anlaß sehen wir uns verpflichtet, unsere Kunden, die SIMATIC-Produkte einsetzen, zu warnen, sogenannte SIMATIC-kompatible Baugruppen fremder Hersteller als Ersatz- oder Zusatzbaugruppen in das Automatisierungssystem SIMATIC einzusetzen.

Unsere Produkte werden einer anspruchsvollen Qualtiätssicherung unterworfen. Uns ist nicht bekannt, ob die fremden Hersteller sogenannter SIMATIC-kompatibler Baugruppen überhaupt oder eine annähernd gleichwertige Qualitätssicherung durchführen. Diese sogenannten SIMATIC-kompatiblen Baugruppen kommen nicht im Einvernehmen mit uns auf den Markt; es gibt **keine** Empfehlung der Siemens AG, sogenannte SIMATIC-kompatible Baugruppen fremder Hersteller einzusetzen. Die Werbung der fremden Hersteller sogenannter SIMATIC-kompatibler Baugruppen erweckt irrtümlich den Eindruck, als sei der Inhalt der Werbung in Fachzeitschriften, Katalogen oder Ausstellungen mit uns abgesprochen. Werden sogenannte SIMATIC-kompatible Baugruppen fremder Hersteller mit unserem SIMATIC-Automatisierungssystem verbunden, handelt es sich um einen empfehlungswidrigen Gebrauch unseres Produkts. Wegen der universellen Vielfalt der Einsatzmöglichkeiten unserer SIMATIC-Automatisierungssysteme und der hohen Zahl der weltweit vermarkteten Produkte, können wir die konkrete Gefahrenanalyse durch diese sogenannten SIMATIC-kompatiblen Baugruppen nicht konkret beschreiben. Es geht über die tatsächlichen Möglichkeiten des Herstellers hinaus, alle diese sogenannten SIMATIC-kompatiblen Baugruppen in ihrer Wirkung auf unser SIMATIC-Produkt überprüfen zu lassen. Treten Mängel bei der Verwendung von sogenannten SIMATIC-kompatiblen Baugruppen in einem SIMATIC-Automatisierungssystem auf, werden wir für solche Systeme jede Gewährleistung ablehnen.

Im Fall von Produkthaftpflichtschäden verursacht durch den Einsatz von sogenannten SIMATIC-kompatiblen Baugruppen sind wir nicht haftbar, da wir die Anwender rechtzeitig vor den potentiellen Gefahren der Benutzung sogenannter SIMATIC-kompatibler Baugruppen gewarnt haben."

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 SIMATICcompatible 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."

Avertissement

Risques liés à l'utilisation de modules de constructeurs tiers commercialisés sous la désignation de "modules compatibles SIMATIC"

≪Le constructeur d'un produit (dans le cas présent SIMATIC) a l'obligation d'observer le produit, c'est-à-dire qu'il est obligé, d'une manière générale, d'attirer l'attention sur les dangers inhérents au produit. Ces derniers temps, la jurisprudence a étendu cette obligation d'observation du produit aux éléments accessoires issus de constructeurs tiers. En foi de quoi, le constructeur a aussi l'obligation d'observer son produit pour déceler les dangers susceptibles de survenir dans le cadre de l'association de son produit avec des produits de constructeurs tiers.

Pour cette raison, nous nous voyons obligés d'attirer l'attention de nos clients, utilisateurs de produits SIMATIC, sur les risques liés à l'utilisation de "modules compatibles SIMATIC" de constructeurs tiers à titre de modules de remplacement ou de complément dans les produits de notre système d'automatisation SIMATIC.

Nos produits font l'objet d'une assurance qualité très poussée. Il nous est impossible de savoir si les constructeurs tiers de "modules compatibles SIMATIC" mettent en œuvre un système qualité et, dans l'affirmative, si leurs dispositions d'assurance qualité permettent d'obtenir le niveau de qualité requis. Les "modules compatibles SIMATIC"ne sont pas commercialisés avec notre consentement ; Siemens AG n'a émis **aucune** recommandation concernant l'utilisation de "modules compatibles SIMATIC" de constructeurs tiers. La publicité des constructeurs tiers de "modules compatibles SIMATIC" laisse penser à tort que les textes publicitaires dans les revues, les catalogues ou les expositions ont été convenus avec nous. L'utilisation conjointe de "modules compatibles SIMATIC" de constructeurs tiers et de produits de notre système d'automatisation SIMATIC constitue un cas d'utilisation de nos produits qui est contraire à nos recommandations. Considérant la grande diversité d'emploi de notre système d'automatisation SIMATIC ainsi que l'importance du parc mondial des produits installés, il nous est impossible de donner une description concrète de l'analyse des risques liés à l'emploi des "modules compatibles SIMATIC". Nous n'avons pas la possibilité matérielle de procéder au contrôle de l'interaction de notre produit SIMATIC avec les "modules compatibles SIMATIC" de constructeurs tiers. Nous rejetons tout appel en garantie pour les vices survenant dans un système d'automatisation SIMATIC mettant aussi en œuvre des "modules compatibles SIMATIC" de constructeurs tiers.

Nous déclinons toute responsabilité pour les sinistres relevant de la Responsabilité Civile Produits, étant donné que nous avons attiré à temps l'attention des utilisateurs sur les risques potentiels inhérents à l'utilisation de "modules compatibles SIMATIC" de constructeurs tiers.≫

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.

Guidelines for Handling Electrostatically Sensitive Devices (ESD)

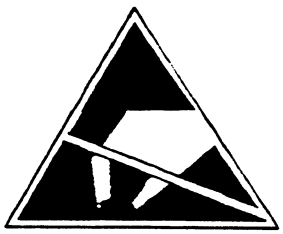
1 What is ESD?

VSLI 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, overstressing 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

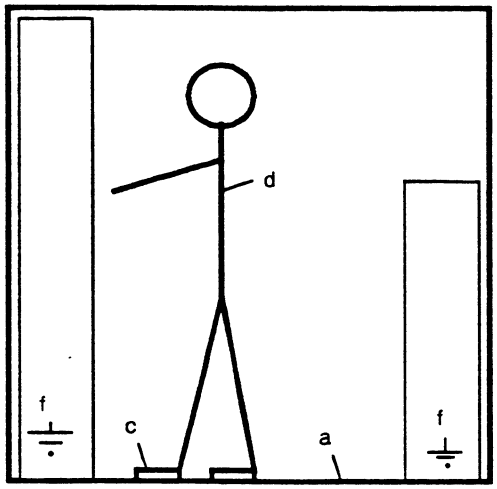
- Most plastic materials are highly susceptible to static charge and must therefore be kept as far away as possible from ESDs.
- Personnel who handle ESDs, the work table and the packing must all be carefully grounded.

4 Handling of ESD Modules

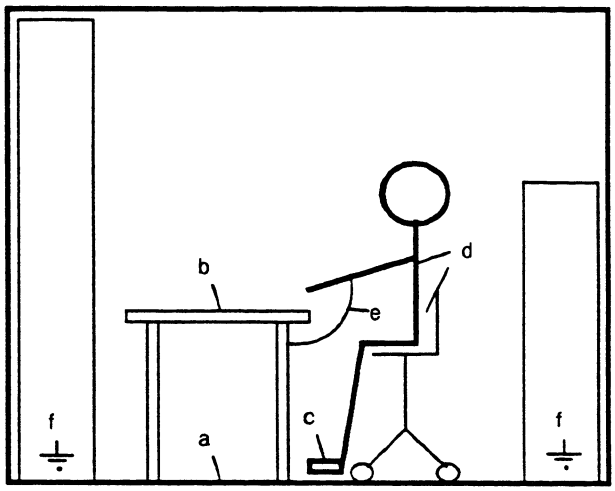
- One basic rule to be observed is that electronic modules should be touched by hand only if this is necessary for any work required to be done on them. Do not touch the component pins or the conductors.
- Touch components only if
  - the person is grounded at all times by means of a wrist strap

  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).

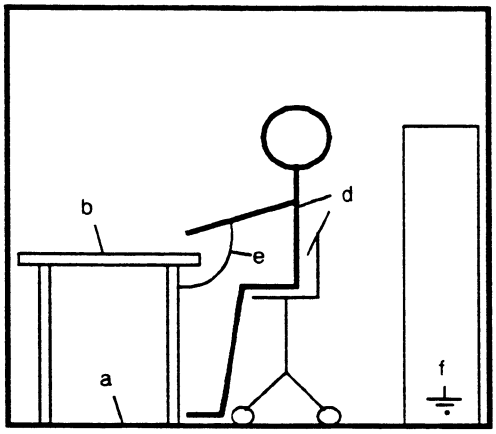
The diagram below shows the required protective measures against electrostatic discharge.



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

S5-135 U Programmable Controller

Data handling blocks for the S processor

Description C79000-B8576-C325-03

Contents Page

1	**General information on the use of data handling blocks**		**2**
1.1	Introduction		2
1.1.1	Configuration		2
1.1.2	Function blocks		4
1.1.3	Memory areas required in the PC		4
1.2	Programmed execution of the data handling blocks		5
1.2.1	Cold restart		5
1.2.2	Synchronizing an interface		5
1.2.3	Queue entries		5
1.2.4	Processing the queue		6
1.3	Data structure of the SSDB		6
1.4	Multiprocessor coordination		7
2	**Assigning parameters to handling blocks**		**8**
2.1	Designations of the parameters		8
2.2	Direct and indirect parameter assignment		9
2.2.1	Parameters SSNR, SSDB, A-NR, ANZW, PAFE, UELA		9
2.2.2	Source and destination parameters		9
3	**Description of parameters**		**12**
3.1	SSNR, SSDB, A-NR, ANZW		12
3.2	Source and destination parameters		14
3.3	Output parameters		17
4	**Evaluation of the output parameters**		**18**
4.1	PAFE, UELA		18
4.2	ANZW		18
4.3	Status		20
4.3.1	Job management on the CP	1st nibble: bits 0 to 3	21
4.3.2	Data management	2nd nibble: bits 4 to 7	22
4.3.3	Error number	3rd nibble: bits 8 to 11	22
4.3.4	Sequence management in the S processor	4th nibble: bits 12 to 15	23
5	**Examples and further information**		**24**
5.1	Synchronizing the interface		24
5.2	Data transfer with DIRECT functions		26
5.3	Background communication with ALL functions		28
5.4	Run times		30
6	**Handling block FB 120: SEND**		**31**
6.1	Block diagram		31
6.2	Description of parameters		31
6.3	SEND, functional description		32
6.3.1	SEND-DIRECT		32
6.3.2	SEND-ALL		33
6.4	SEND, operational sequence		34
7	**Handling block FB 121: RECEIVE**		**35**
7.1	Block diagram		35
7.2	Description of parameters		35
7.3	RECEIVE, functional description		36
7.3.1	RECEIVE-DIRECT		36
7.3.2	RECEIVE-ALL		37
7.4	RECEIVE, operational sequence		38
8	**Handling block FB 122: FETCH**		**39**
8.1	Block diagram		39
8.2	Description of parameters		39
8.3	FETCH, functional description		40
8.4	FETCH, operational sequence		40
9	**Handling block FB 123: CONTROL**		**41**
9.1	Block diagram		41
9.2	Description of parameters		41
9.3	CONTROL, functional description		42
9.3.1	CONTROL-DIRECT		42
9.3.2	CONTROL-ALL		42
9.4	CONTROL, operational sequence		43
10	**Handling block FB 124: RESET**		**43**
10.1	Block diagram		43
10.2	Description of parameters		43
10.3	RESET, functional description		44
10.3.1	RESET-DIRECT		44
10.3.2	RESET-ALL		44
10.4	RESET, operational sequence		44
11	**Handling block FB 125: SYNCHRON**		**45**
11.1	Block diagram		45
11.2	Description of parameters		45
11.3	SYNCHRON, functional description		46
11.4	SYNCHRON, operational sequence		46
12	**Handling block FB 126: ACTIVE FB 127: UP ACTIV**		**47**
12.1	Block diagram		47
12.2	Description of parameters		47
12.3	ACTIVE, functional description		48
12.4	ACTIVE/UP ACTIV, operational sequence		48
13	**List of keywords**		**49**

1 General information on the use of data handling blocks

1.1 Introduction

The Siemens SIMATIC S5 range is a system of electronic modules and devices for the automation of industrial plant. The SIMATIC S5 electronic modules include processors which are used to perform a variety of tasks such as open and closed loop control, operator-process communication (operation) and monitoring in industrial processes. Several such processors are installed in one **programmable controller (PC)**.

The following SIMATIC S5 processors are available:

- **Central processing units (CPU)**. These are the central processors of a programmable controller. They execute the user programs written in the STEP 5 programming language.

- **Communications processors (CP)**. These are installed as front-end processors and are used for operator-process communication and process monitoring (e.g. CP 526).

- **Intelligent I/O modules (IP)**. These are also installed as front-end processors and handle more complex automation tasks, e.g. positioning module IP 246.

Information is exchanged between the CPU's, the communications processors and the intelligent I/O modules using **data handling blocks**.

These **operating instructions** describe the **handling blocks** for the CPU's **S processor 6ES5921-3UA11** and **6ES5921-3UA12** in the S5 135 U programmable controller.

In order to simplify matters the term CP will be used both for communications processors and intelligent I/O modules since there is no difference between them with respect to data handling blocks.

1.1.1 Configuration

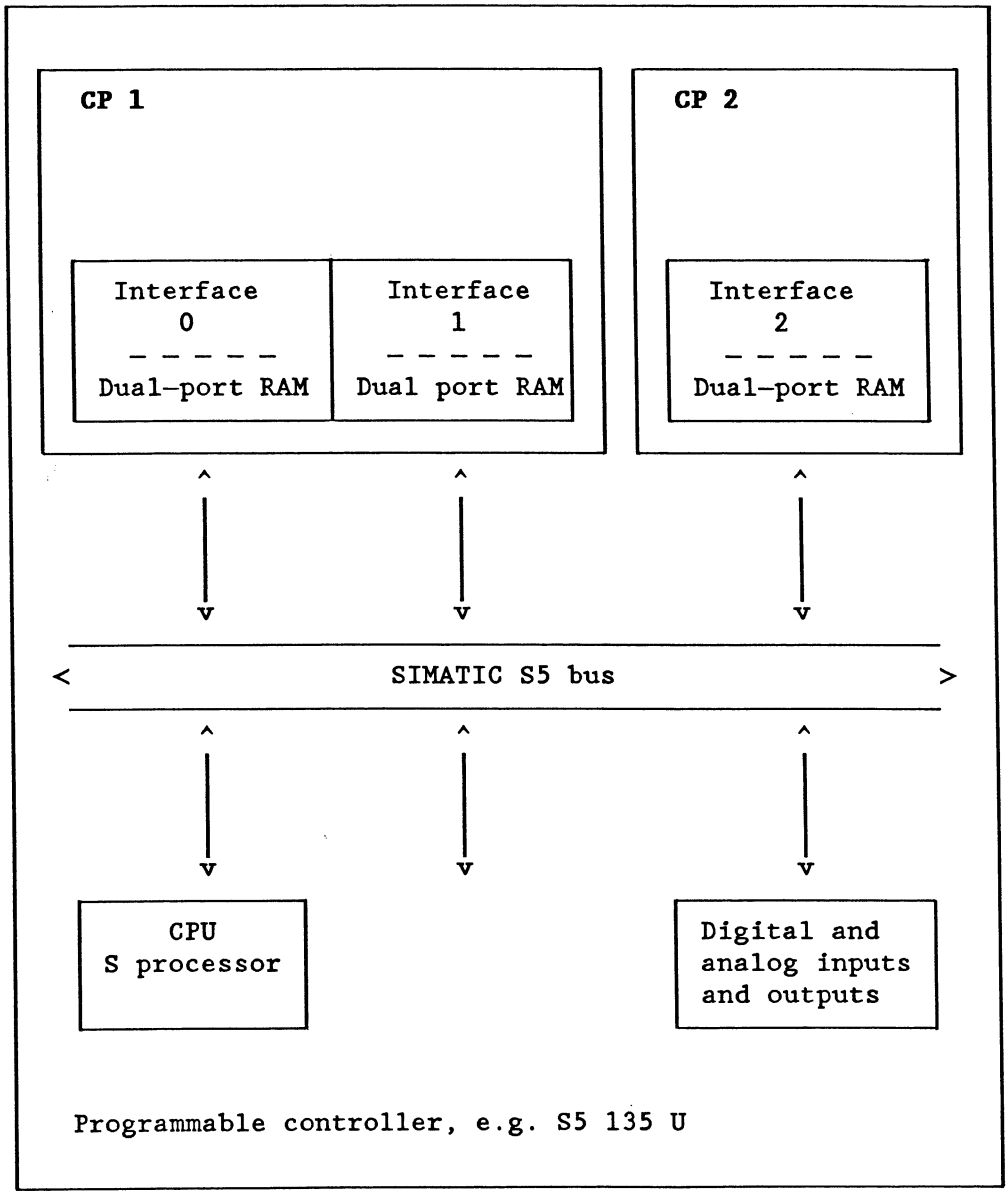
Each CP has a special memory submodule known as a **dual-port RAM**. If page addressing is being used, the dual-port RAM is often described as a **page**. This memory is used for the exchange of information between the CPU and the CP.

The CPU of a PC accesses (reads from or writes to) the dual-port RAM of a CP via the SIMATIC S5 bus which is located on the rear panel of the PC. The microprocessor of a CP accesses its own dual-port RAM (writing or reading) in exactly the same way.

The exchange of information between the two partners CPU and CP always uses the same basic strategy, i.e. one partner (e.g. the CPU) writes its information to the dual-port RAM of the CP and the other partner (e.g. the microcomputer of the CP) reads this information from the dual-port RAM and processes it further.

The dual-port RAM itself, as well as the hardware and software facilities which write information to or read information from the dual-port RAM, and which establish the link to the actual CP function, will simply be called **interface** from now on. An interface is selected on the CP with the parameter of the interface number **SSNR** (see chapter 3). Depending on their type, the CP's can have one or several dual-port RAM's (interfaces).

Example: possible configuration with one CPU and two CP's with a total of three interfaces.



The functions of a CP are designated as so-called **jobs**. With the CP 526, for example, a typical job would be the display output on a monitor. Each job has a job number **A-NR** assigned to it when the CP is programmed (see programming package for CP 526: COM 526 manual). By using the data handling blocks such jobs can be manipulated (start, monitor, execute, abort and finish). With the CP 526 for instance, the job number can be used to start the output of a particular display on the monitor.

1.1.2 Function blocks

Data, parameters and control information are exchanged between CPU's and CP's by means of **data handling blocks**, which are entered in a **queue** until they are processed. The following function blocks are available:

Name	FB	Function of the function block call
SEND	FB120	Request to CPU to **send data** to the CP; request is entered in the queue
RECEIVE	FB121	Request to CPU to **receive data** from the CP; request is entered in the queue
FETCH	FB122	Request to CP to **fetch data** for the CPU; request is entered in the queue
CONTROL	FB123	Request to CPU to **update the status**; request is **not** entered in the queue
RESET	FB124	Request to CP to **reset a job**; request is entered in the queue
SYNCHRON	FB125	Request to **initialize queue** and **allocate interface**; request is entered in the queue
ACTIVE	FB126	Request to **process queue entries**; request is not entered in the queue
UP ACTIVE	FB127	**Subroutine** of the ACTIVE function block, only called by the ACTIVE block

By means of the data handling block CONTROL the user can find out which job is currently running at any time on a particular CP and can determine the current processing STATUS of a particular job.

The queue entries are processed by repeatedly calling the ACTIVE block with its subroutine UP ACTIV.

1.1.3 Memory areas required in the PC

The data handling blocks require the following fixed memory areas in the PC:

- the flag area from FB 200 to FB 255,
- the system data area BS 254,
- one data block per interface in order to set up the queue.

Depending on the program, the data handling blocks also access various flags, data words and other memory areas in the PC, which are identified by the parameters referenced when the data handling blocks are called.

1.2 Programmed execution of the data handling blocks

1.2.1 Cold restart

With each of the restart modes of the S processor

- with OB 20: manual cold restart with reset,
- with OB 21: manual cold restart retaining flags,
- with OB 22: automatic cold restart retaining flags

it is advisable to synchronize all the interfaces (in the restart OB) which will later be used to transfer data by means of the data handling blocks. Only when these interfaces have been synchronized can the corresponding data handling blocks be called up in the program.

Chapter 5 contains examples of the required programming steps. It is, however, also advisable to refer to the programming examples in the manuals for the specific CP's. These programming examples may differ from those in chapter 5 owing to the characteristics of the specific CP.

1.2.2 Synchronizing an interface

Each interface must first be initialized by means of the data handling block **SYNCHRON**. SYNCHRON ought to be called during the restart in the restart OB's (OB 20, OB 21 and OB 22).

When it is called SYNCHRON sets up the data block **SSDB** as a queue block which is allocated to the interface SSNR. This at the same time initializes the queue and SYNCHRON enters itself in the queue as the first call. This entry is processed by means of further ACTIVE block calls; this includes among other things the clearing and preassignment of the dual-port RAM. SYNCHRON is then completed and is deleted from the queue.

1.2.3 Queue entries

Each time a data handling block (SYNCHRON, SEND, RECEIVE, FETCH, RESET) is called an entry is made in the queue. If the queue block SSDB is full then no further handling block calls can be entered. A queue overflow is signalled to the user by setting a bit. When programming, the user should avoid entries for the same request being entered repeatedly in the queue, in order to ensure sufficient space for other entries.

1.2.4 Processing the queue

By repeatedly calling the data handling block ACTIVE the queue is processed and the required exchange of information executed. For this reason ACTIVE is called in the cyclic part of the user program, i.e. in OB 1 or in FB 0. Calling ACTIVE cyclically ensures that the queue is processed quickly.

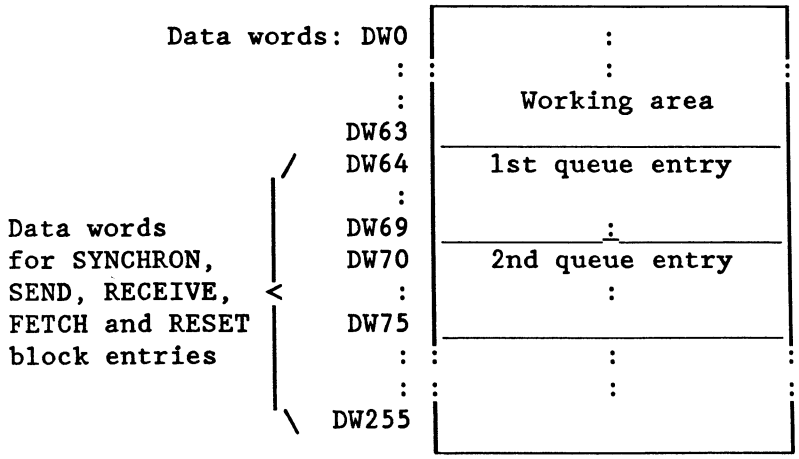
A special handshaking procedure is used to coordinate the exchange of information so that a variety of data and parameters can be transferred in both directions via one interface.

1.3 Data structure of the SSDB

When using data handling blocks one SSDB data block must be established per interface (see section 1.2.2). This does not need to be preassigned or called by the user.

The data block consists of a 64 word long working area and a queue area of variable length. Each entry in the queue requires 6 words. The minimum length of the data block is therefore 70 words. words. The actual length of the queue is determined by the number of entries it is required to accept. The maximum length of the data block is 256 words.

Structure of the interface data block **SSDB**:



1.4 Multiprocessor coordination

Up to 4 CPU's can operate simultaneously in one S5 135 U programmable controller, with a coordinator allocating the SIMATIC S5 bus for a certain length of time to each CPU in succession, to allow it to transfer data. This configuration is known as a SIMATIC S5 multiprocessor system.

If the multiprocessor system consists of only S processors, care must be taken that handling blocks are called at most in the program of one S processor. If the program of more than 1 S processor calls handling blocks then problems will occur owing to addresses overlapping.

If the system consists of various processor types (S, R and M processors) then one S processor or each of the R and M processors can use the handling blocks.

For this reason the **S processor handling blocks** in the S5 135 U programmable controller **do not** have **multiprocessing capability.**

2 Assigning parameters to handling blocks

2.1 Designations of the parameters

Whenever a handling block is called in the user program, **parameters** are specified. These block parameters have the following uniform designations and fixed order:

– **SSNR**	Interface number, which is already fixed for the interface in the CP by means of plug–in jumpers
– **SSDB**	Number of the queue block, which is allocated to the corresponding interface SSNR
– **A–NR**	Job number, which identifies a particular CP job
– **ANZW**	Condition codeword belonging to a job A–NR on the CP and which indicates the current processing status of this job to the user
– **QTYP/ZTYP**	Type of data source/data destination
– **DBNR**	Number of the data block in the PC from which data or further block parameters are taken or in which data will be stored.
– **QANF/ZANF**	Start address of the first data word to be transferred or the first "actual" parameter in the DBNR data block
– **QLAE/ZLAE**	Length of source/destination data block (number of data words) which will be read from/written to the DBNR
– **PAFE**	Condition code bit for parameter assignment errors
– **UELA**	Condition code bit for queue overflow

Each handling block has a specific selection of these parameters which must be assigned current values by the user when the block is called. There are, however, functions which do not require all the parameters. All the parameters must, nevertheless, be supplied with values, which means that in some cases irrelevant values must be specified (simply to avoid leaving gaps).

2.2 Direct and indirect parameter assignment

With **direct** parameter assignment the handling block immediately processes the parameters specified in the block call. By specifying direct parameters the handling block can trigger a CP job immediately when it is processed by ACTIVE. If the particular job also involves a data transfer between the CPU and CP this is also carried out directly via the corresponding CP interface.

With **indirect** parameter assignment the handling block is initially supplied with a pointer to a parameter field in a data block of the CPU by specifying certain source or destination parameters when it is called up in the user program. By means of this pointer the handling block finds the "actual" source or destination parameters stored in the same order as they would be specified with direct parameter assignment. Only when the handling block is aware of the actual (direct) parameters is it capable of triggering a job on the CP including any data transfer which may be required.

A detailed description of all the block parameters can be found in chapter 3; the evaluation of certain parameters (output parameters) is described in chapter 4.

2.2.1 Parameters SSNR, SSDB, A-NR, ANZW, PAFE, UELA

These parameters must be specified by the user of the handling blocks (directly or indirectly) when the block is called in the program.

2.2.2 Source and destination parameters

With **direct** parameter assignment the handling block processes the source and destination parameters specified in the block call directly.

The **source** parameters QTYP, DBNR, QANF, QLAE identify a memory area in the CPU as a source of data which can be transferred directly to the CP interface.

The **destination** parameters ZTYP, DBNR, ZANF, ZLAE also identify a memory area of the CPU which is the destination for data to be stored when they are transferred via the CP interface. In this procedure the "old" data are overwritten.

Source and destination parameters on the block with direct parameter assignment:

Parameter	Meaning	Param. type	Data type	Example
QTYP/ZTYP	Source/dest. type	D	KC	KSDB (direct)
DBNR	Data block number stored in low byte	D	KY	KY0,18 (DB 18)
QANF/ZANF	Source/dest. start: start address of first data word in corresponding DB	D	KF	KF+3 (from DW3)
QLAE/ZLAE	Source/dest. length: no. of data words to be transferred	D	KF	KF+6 (6 DW's)

With **indirect** parameter assignment the specified source or destination parameters refer to a **parameter field** in a data block which contains the "actual" source or destination parameters. If the relative start address QANF/ZANF points to e.g. the data word DW n of the corresponding data block DBNR then the 4 successive data words: DW n to DW n+3 are the actual source/destination parameters.

Storing the "actual" source and destination parameters in the DBNR with indirect parameter assignment:

DW in the DBNR		Format	Parameter	Example	Hex
:					:
:					:
DW n \		**KS** (ASCII)	QTYP/ZTYP	DB	4442
DW n+1 |	Para-	**KY** (2 bytes)	DBNR	177	00B1
>	meter				
DW n+2 |	field	**KF** (fixed point)	QANF/ZANF	50	0032
DW n+3 /		**KF** (fixed point)	QLAE/ZLAE	30	001E
:					:
:					:

The parameter field in the example above indicates that when the handling functions SEND/RECEIVE are processed the data transfer will be from/to the data block DB 177 and that the 30 data words from data word 50 are involved.

For the parameters QTYP/ZTYP the following types can be specified in the block: DB, XX, SS, NN. Depending on the type specified different functions will be triggered when the handling blocks are processed:

For **direct parameter assignment** the parameter QTYP/ZTYP must be specified as **DB**. With this specification, all the subsequent source or destination parameters (DBNR, QANF/ZANF and QLAE/ZLAE) are relevant and are evaluated when the block is called since these three parameters are required to define a memory area (block of data).

- For **indirect parameter assignment** the QTYP/ZTYP must have **XX** or **SS** specified. In both cases (XX and SS) the two subsequent source/destination parameters DBNR and QANF/ZANF serve as a pointer to a memory field which contains the four actual source or destination parameters. The specification of the parameters QLAE/ZLAE in the block is then irrelevant (but must be made, e.g. KF+0), since the block of data with the actual source/destination parameters is always 4 data words long.

- With the type **XX** the reading of the actual parameters is immediately followed by a data transfer when the block is processed, while with type **SS** only the actual parameters are transferred to the CP; the final transfer of data only takes place in this case when a corresponding ALL function is called during the processing of the queue.

- With type **NN** only the job number A-NR specified on the block is relevant. When it is called the handling block triggers a job on the CP which may also include a data transfer. The source/destination parameters belonging to the particular job number A-NR are supplied by the CP.

3 Description of parameters

3.1 SSNR, SSDB, A-NR, ANZW

- **SSNR**: interface number

  The number of an interface on a CP with which the CPU e.g. is to exchange information. The interface will only be enabled for the exchange of information with the S5 bus on the CP on which this number (address) is set by means of plug-in jumpers. The parameter SSNR must be specified for the interface when calling SYNCHRON or CONTROL.

  Parameter type/data type: data/KY (KY ≙ 2 bytes)
  permissible range : 0 ... 255, 0 ... 255
  recommended range : **0**, 0 ... 255 (high, low byte)

  Only the low byte is evaluated; this contains the interface number. The information in the high byte is irrelevant; usually contains 0.

- **SSDB**: interface data block

  The data block which is initialized as the queue block and allocated to the interface SSNR during the SYNCHRON call. Its entries are handling block calls which are processed in the order in which they are entered by repeatedly calling ACTIVE.

  Parameter type/data type : B/- (B ≙ command)
  permissible range : DB 3 ... DB 255

  DB 0, DB 1 and DB 2 are reserved for special purposes.

- **A-NR**: job number

  Each job number identifies a particular job on the interface which must be programmed on the CP. The job is triggered on the CP by a handling block.

  Parameter type/data type: data/KY (KY ≙ 2 bytes)
  permissible range : 0 ... 255, 0 ... 223
  recommended range : 0, **0** (ALL. function)
  recommended range : 0, 1 ... 199 (DIRECT function)

  Only the low byte is evaluated, it contains the job number. The information in the high byte is irrelevant; normally contains 0.

  The job numbers from 200 onwards are mainly reserved for system jobs which have a special significance for all CP's.

  If **low byte = 0** is specified for the job number **A-NR** then this is an **ALL function**. This function checks whether the interface has a communication request. If there is a communication request, the handling block reads the corre-

sponding job number A-NR and the source and destination parameters for the data transfer from the interface and uses them to carry out the data transfer.

The ALL function remains entered in the queue until all the outstanding communication requests on the interface have been serviced (if necessary, by means of several ACTIVE calls). The handling block is informed of this by the interface.

If the **low byte ≠ 0** then a **DIRECT function** is present. In this case the data transfer is carried out under the A-NR and using the source and destination parameters as specified in the handling block call.

- **ANZW**: condition codeword

For every job i.e. for every defined job number a condition codeword should be made available with the parameter ANZW. This allows the user to follow the processing of any job.

Parameter type/data type: data/KY (KY ≙ 2 bytes)
permissible for flag words: **0** , 0 ... 198
permissible for data words: 3 ... 255, 0 ... 255

The low byte of ANZW always contains the number of a flag or data word.

If the **high byte = 0** this indicates that the condition codeword is stored in a **flag word**.

If the **high byte ≠ 0** this indicates that the condition codeword is stored in a **data word**, the high byte specifying the number of the data block in which the data word is located. Since DB 1 and DB 2 are reserved for special functions only data blocks from DB 3 onwards are available for the ANZW.

The blocks which affect the condition codeword and how it can be evaluated are described in detail in chapter 4 ("Evaluating the output parameters").

3.2 Source and destination parameters

The source and destination parameters specified in the block are only evaluated when a job is being processed if a DIRECT function (A-NR ≠ 0) is present.

If A-NR = 0 (i.e. ALL function) the specified source and destination parameters are irrelevant; the actual source/destination parameters will then be supplied by the CP along with the corresponding job number A-NR.

- **QTYP/ZTYP**: type of data source/data destination

  With this parameter ASCII characters can be used

  - in the SEND block call to specify the type of data source,
  - in the RECEIVE and FETCH block calls to specify the type of data destination.

  Parameter type/data type: data/KC (≙ ASCII characters)
  permissible entries : DB, XX, SS, NN

  - direct parameter assignment : DB
  - indirect parameter assignment: XX, SS
  - parameter supplied by CP : NN

  If the **DB** type is used, as soon as the handling block is called in the queue it triggers the data transfer between the CPU and the CP interface immediately using the parameters specified in the block.

  If the **XX** type is used, then the parameter assignment is indirect. The two following parameters DBNR and QANF/ZANF serve as a pointer to a parameter field in a data block DBNR in which the four actual source or destination parameters are located from the relative start address QANF/ZANF. The parameter QLAE/ZLAE must also be specified although it is irrelevant. The block of data defined by the four actual parameters is also transferred when the block is called.

  If the **SS** type is used, the four actual parameters will be read by means of the two indirect parameters DBNR and QANF/ZANF, however, they will initially only be stored on the CP along with the job number A-NR. The block of data which they define is transferred later when a corresponding ALL function is called in the queue.

  If the **NN** type is used, the following parameters DBNR, QANF/ZANF and QLAE/ZLAE are irrelevant. The CP then provides the actual source/destination parameters corresponding to the specified job number A-NR for the handling block, which then triggers the data transfer using these parameters.

- **DBNR**: data block number

  The data block number DBNR is only evaluated when the parameter QTYP/ZTYP is of type DB, XX or SS. If NN is specified the following source or destination parameters are not evaluated.

  Parameter type/data type: data/KY (KY ≙ 2 bytes)
  permissible range : 0 ... 255, 3 ... 255
  recommended range : **0** , 3 ... 255

  Only the low byte is evaluated. This specifies the data block in which the data to be transferred are located when using direct parameter assignment (type: DB); with indirect parameter assignment (type: XX, SS) they specify the block in which the four actual source or destination parameters are located.

- **QANF/ZANF**: start address of the source/destination block of data

  QANF/ZANF is the start address of the first data word of the source/destination data in the data block DBNR, which are to be transferred when using direct parameter assignment (QTYP/ZTYP = DB).

  With indirect parameter assignment (QTYP/ZTYP = XX, SS) QANF/ZANF is also the start address of a block of data in data block DBNR. This block however only consists of the four data of the actual source or destination parameters.

  Parameter type/data type: data/KF (KF ≙ fixed point no.)
  permissible range : +0 ... +255

- **QLAE/ZLAE**: length of the source/destination data block

  The parameters QLAE/ZLAE specify the number of data words to be transferred from a data source or to a data destination when using direct parameter assignment (type = DB).

  Parameter type/data type: data/KF (KF ≙ fixed point no.)
  permissible range : +1 ... +256

  When using indirect parameter assignment the values specified for QLAE/ZLAE are irrelevant since exactly four data words are read as the actual source/destination parameters.

The dependence of the source and destination parameters on the specified QTYP/ZTYP (DB, XX, SS or NN) and the various functions of the handling blocks are summarised in the following tables:

Source/ dest. type **QTYP/ ZTYP**	Description	Data block number **DBNR**	Source start/ dest. start **QANF/ ZANF**	Source length/ dest. length **QLAE/ ZLAE**
DB	source/ dest. data taken direct from block/ stored direct in block	block from which data are taken/ in which data are stored	address of DW from which data words are taken/ stored	length of block of data in words
XX	indirect param. ass. with data exchange; actual s/d params. in data block	block in which act. source/ destination parameters are stored	address of DW from which the 4 actual s/d param. are stored	irrelevant
SS	indirect param. ass. without data exchange; * actual s/d params. in data block	block in which act. source/ destination parameters are stored	address of DW from which the 4 actual s/d param. are stored	irrelevant
NN	CP supplies actual s/d parameters acc. to job no. on block A–NR > 0 followed by data transf.	irrelevant	irrelevant	irrelevant

* When using the type SS the data transfer is carried out later by means of an ALL function.

3.3 Output parameters

The output parameters are those handling block parameters which are influenced (changed) while the handling block functions are being executed allowing the user to follow the course of the processing. The output parameters are:

- PAFE: parameter assignment error
- UELA: overflow (of the queue)
- ANZW: condition codeword

While PAFE and UELA are pure output parameters, ANZW has a bit which is not set by the handling block but rather by the user. This bit retains the value that it had when the handling function was called. A difference is therefore made between the input ANZW and output ANZW.

- **PAFE**: parameter assignment error

  The output parameter PAFE provides information about the processing of a SYNCHRON or CONTROL block.

  If a parameter assignment error is present when a SYNCHRON is called (e.g. SSDB not specified or not long enough) the PAFE bit will be set and the block exited. The condition codeword is not changed nor is a queue initialized.

  If a parameter assignment error is present when a CONTROL is called (e.g. specified interface number SSNR cannot be found) the block will be exited with PAFE = 1.

  Parameter type/data type: output/bit
  permissible range : I, Q, F 0 ... 255 . 1 ... 7

- **UELA**: queue overflow

  With the handling blocks which are entered in the queue, the output parameter UELA provides information as to whether the entry of the next call in the queue would cause an overflow.

  Parameter type/data type: output/bit
  permissible range : I, Q, F 0 ... 255 . 1 ... 7

- **ANZW**: condition codeword (see section 3.1 and section 4.2)

4 Evaluation of the output parameters

4.1 PAFE, UELA

Both parameters consist of only one bit.

- **PAFE** is only affected when a SYNCHRON or a CONTROL is called and indicates a parameter assignment error.

  If, e.g., when calling a SYNCHRON an SSDB is specified as the queue block which is either not present or not long enough then no queue can be set up; this error is indicated by PAFE = 1.

  PAFE = 1 is also indicated when a CONTROL is called if an illegal interface or job number has been specified in the block.

- **UELA** is affected by the handling blocks which are entered in the queue when they are called in the user program, i.e. SEND, RECEIVE, FETCH and RESET. UELA indicates whether there is enough space in the queue block SSDB for a handling block to be entered (UELA = 0) or whether the entry of this handling block will cause an overflow (UELA = 1). If an overflow would result, the handling block cannot be entered in the queue; the user must take this into account when programming (troubleshooting).

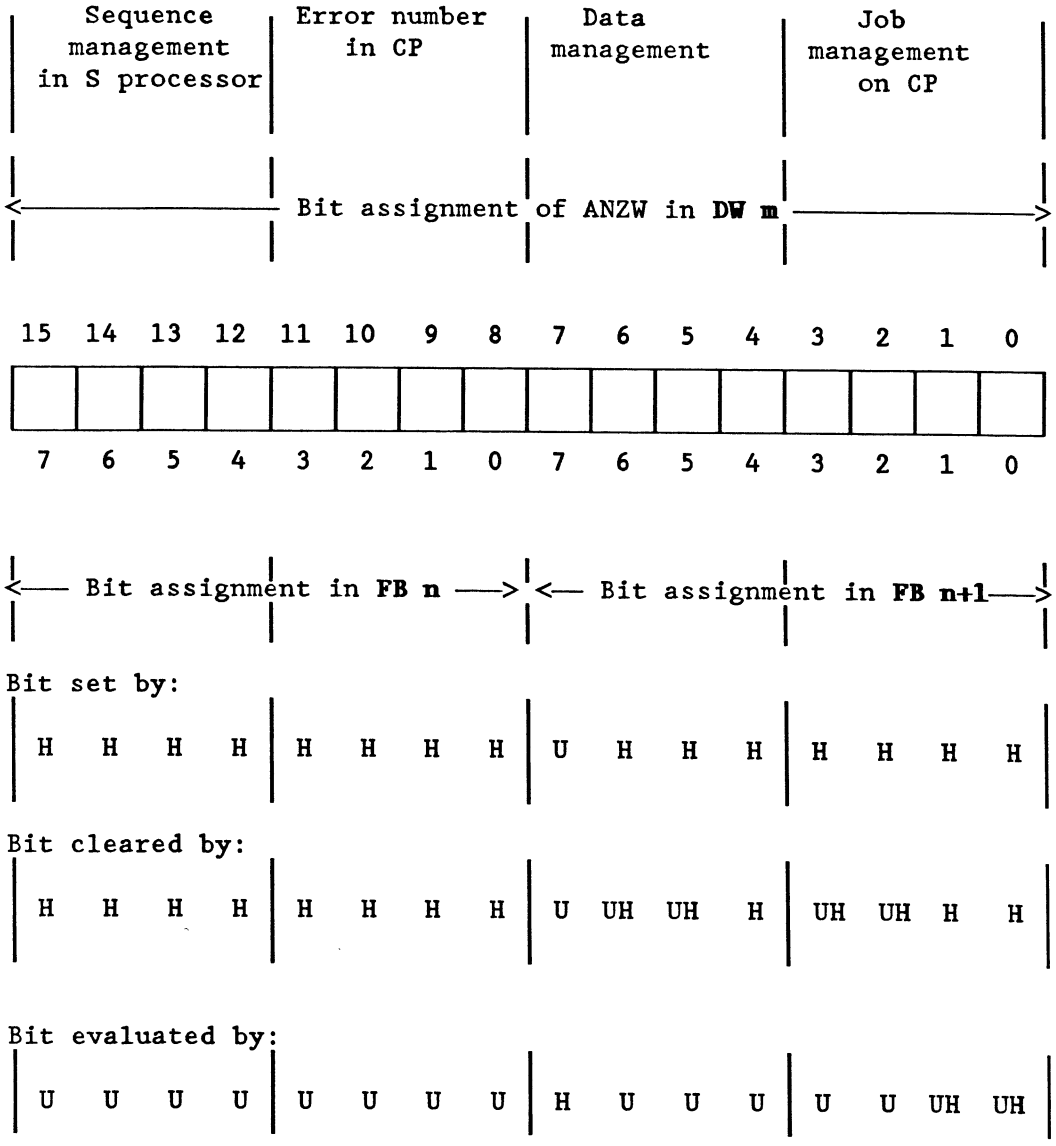
4.2 ANZW

The ANZW condition codewords are used to monitor the processing status of jobs. It is advisable to allocate a condition codeword to each job. This allocation of condition codewords to jobs is carried out when the handling block is called by specifying the job number A-NR and the address of the condition codeword ANZW. The condition codeword of a job can be changed several times by the CP or the handling block while a job is being processed, allowing the user to follow the processing. The condition codewords are stored in data or flag words, the latter being preferable owing to the better run time.

The condition codeword specified in the block is designated ANZW or ANZW-AG. It contains the STATUS information (exception: CONTROL-ALL).

STATUS provides information about the job management in the CP, data management, the error number in the CP, the sequence management in the S processor. These four terms are explained in more detail in section 4.3.

Storing the **ANZW-AG** in data word DW m or in flag word FW n:



H = handling block sets, clears, evaluates
U = user sets, clears, evaluates

In some cases the CP has to take over the ANZW address specified at the block call. This occurs e.g. when the processing of a job takes longer than the processing of the handling block which started the job.

When this situation arises, the address of the ANZW-AG is stored in the CP. From this point onwards the corresponding condition codeword is called **ANZW-CP** because its address is now supplied by the CP. The ANZW-CP is now updated as events occur in the processing of the job.

The condition codeword assignment according to the handling block used is shown in the following table:

Type of job	ANZW–AG	ANZW–CP
SEND–DIRECT/RECEIVE–DIRECT	Status	–
SEND–ALL/RECEIVE–ALL	Status	Status
FETCH (DIRECT)	Status	–
RESET–DIRECT/RESET–ALL	Status	–
CONTROL–DIRECT	Status	–
CONTROL–ALL	A–NR	–

With SEND-ALL and RECEIVE-ALL the ANZW-CP is only written to if the CP supplies a permissible ANZW address when the block is called and a data transfer takes place.

With some functions parts of the ANZW are irrelevant:

- with SEND/RECEIVE-ALL : bits 0-3 and bits 8-11
- with RESET-ALL : bits 0-11
- with RESET-DIRECT : bits 4-7
- with SYNCHRON : bits 0-11

these parts of the ANZW are irrelevant.

4.3 Status

When the handling blocks SEND, RECEIVE, FETCH, RESET and CONTROL-DIRECT are processed the STATUS is entered in the condition codeword ANZW-AG which provides information about the status of:

- job management in the CP = 1st nibble in ANZW = bits 0-3
- data management = 2nd nibble in ANZW = bits 4-7
- error number from CP = 3rd nibble in ANZW = bits 8-11
- sequence management in S proc.= 4th nibble in ANZW = bits 12-15

In each CP interface there is a **job status** for each job. A specific memory area is reserved for this status. The job status is managed by the interface itself and contains two areas each with 4 bits for job management and error number. The **job management** indicates whether handshaking is feasible, whether a certain job in the CP is (still) running or whether it was completed with or without errors. If the job was completed with errors the **error number** indicates which error occurred during the processing. While a handling block is being processed the 4 bits from the job management and the 4 bits of the error number are copied into the 1st and 3rd nibbles of the ANZW respectively.

By means of the 4 **data management** bits, handling blocks and the user can check whether a data transfer is running or whether a SEND or RECEIVE job has already been completed. The user (not the handling block) can also inhibit the transfer of the corresponding data area by setting bit **7**.

The 4 **sequence management** bits in ANZW-AG indicate whether the corresponding job A-NR is still entered in the queue or is currently being processed by the handling block or whether the handling block processing is already complete.

4.3.1 Job management on the CP 1st nibble: bits 0 to 3

The bits copied into the ANZW from the CP job status while a handling block is being processed have the following significance:

- **Bit 0: handshake feasible**
  Bit 0 = 1 indicates to the handling block and the user that a handshaking procedure is feasible, since data for the CPU are available on the CP for a RECEIVE call.

- **Bit 1: job running**
  Bit 1 = 1 indicates to the user and the handling block that a job is currently running in the CP and that no new job can be accepted at present. The CP can only accept a new job from the handling block when the old job has been completed (bit 1 = 0).

- **Bit 2: job complete without errors**
  Bit 2 = 1 indicates to the user that the job was completed by the interface without errors.

- **Bit 3: job complete with error**
  Bit 3 = 1 indicates to the user that the job was completed by the interface but that an error occurred. The cause of the error is indicated by bits 8 to 11 (error number).

Examples of the 1st nibble of the ANZW:

a) Bit 3 2 1 0 Significance for SEND and FETCH blocks:

= **x x 0 x** CP ready for this job (x = 1 or 0)
= **x x 1 x** CP not ready for this job

b) Bit 3 2 1 0 Significance for the RECEIVE block:

= **x x x 1** Data available on CP for handshake and CP ready for this job
= **x x x 0** No data available on CP for handshake and CP not ready for this job

c) Bit 3 2 1 0 Significance for SEND, RECEIVE, FETCH jobs:

= **0 1 0 0** job completed on CP without errors
= **1 0 0 0** job completed on CP with error

4.3.2 Data management — 2nd nibble: bits 4 to 7

With bits 4 to 6 the user can check whether a data transfer is currently running. With bit 7 the user can inhibit the data transfer for a job.

- **Bit 4: data transfer running**
  Bit 4 = 1 indicates to the user that a data transfer is currently running.

- **Bit 5: data transfer completed**
  Bit 5 = 1 indicates to the user that data have been transferred to the CP following a SEND job.

- **Bit 6: data acceptance completed**
  Bit 6 = 1 indicates to the user that the CPU has accepted the data following a RECEIVE job.

- **Bit 7: inhibit block of data**
  If bit 7 = 1 is set by the user this indicates to the handling block that the transfer of the data for this job is inhibited: in this case the function is aborted.

4.3.3 Error number — 3rd nibble: bits 8 to 11

The CP enters a previously declared hexadecimal error number in these 4 bits of the ANZW if the job was completed with an error (ANZW bit 2 = 0 and ANZW bit 3 = 1).

The error numbers have the following significance:

error no. 0:	no error
error no. 1 to 5:	errors recognised by the handling functions
error no. 1 to 4:	source/destination area not permissible/not available
error no. 5:	error in ANZW address
error no. 6 to F:	errors recognised by the CP's.

4.3.4 Sequence management in the S processor

4th nibble: bits 12 to 15

These 4 bits of the ANZW indicate to the user whether and how the handling block has been processed.

- **Bit 12: completed with error**
  Bit 12 = 1 indicates to the user that the handling function has been completed but with an error or has been aborted, e.g. owing to a parameter assignment error A-NR > 223 or because of a fault at the CP.

- **Bit 13: no processing possible**
  Bit 13 = 1 indicates to the user that the handling function could not be processed when it was called, since e.g. the interface was overloaded or the transfer of the block of data was inhibited. In order to remedy the error the function should be called again, if necessary several times.

- **Bit 14: entered in queue**
  Bit 14 = 1 indicates to the user that the handling function has been entered in the queue.

- **Bit 15: processing started**
  Bit 15 = 1 indicates to the user that the processing of the handling function has begun.

Examples of the 4th nibble of the condition codeword:

Bit	15	14	13	12	bit combinations have the following significance:
=	0	0	0	0	handling block not entered in queue
=	0	1	0	0	handling block entered in queue
=	1	1	0	0	handling function already started
=	0	0	0	0	handling function complete without error
=	0	0	0	1	handling function complete with error
=	0	0	1	0	handling function could not be executed

The combination of bits 12 to 15 = 0 can therefore mean that the handling function has not yet been entered in the queue or that it has been processed already without an error and has been deleted from the queue.

Bits 0 to 13 of the condition codeword must not be evaluated as long as bit 15 = 1. Once the handling function has been completed, bit 15 will be cleared again; the job will be deleted from the queue and bit 14 will be cleared at the end of the job.

5 Examples and further information

Handling blocks can be called at any point in the user program; e.g. in the organization blocks

- for restart OB 20, OB 21, OB 22,
- for cyclic operation OB 1, FB 0,
- for time-controlled operation OB 13,
- for interrupt-driven operation OB 2.

When planning and designing a system the priorities with respect to reciprocal interruptibility of the organization blocks must be taken into account.

Handling block calls are of course also possible in program blocks PB's and in other function blocks FB's.

In the following sections the use of handling blocks will be illustrated using typical examples. It is assumed that the user has cleared the condition codewords, event flags and result flags in the restart OB's.

5.1 Synchronizing the interface

Before using handling blocks the CPU and the CP must be synchronized. This is carried out in the restart OB's (OB 20 to OB 22) by calling a SYNCHRON per interface.

The SYNCHRON initializes the data block SSDB specified in the SYNCHRON call as a queue, adds the corresponding interface number SSNR and enters itself as the first block in the queue.

In the same restart OB, ACTIVE blocks can also be called. By means of these blocks the SYNCHRON in the queue is further processed and the interface (dual-port RAM) is initialized. Only after the SYNCHRON has been processed without errors and the CP interface initialized is it feasible to call handling blocks.

Example 1: SYNCHRON

An interface is to be synchronized on a CP for which the interface number 4 has been set at the CP. DB 14 is to be used as the queue DB, FW 6 as the condition codeword and F 1.0 as PAFE. DB 14 already exists and is long enough. OB 20 has been selected as the start-up OB.

Program section in OB 20:

```
        :
        :JU  FB10
   NAME :START-UP
```

Program in FB 10:

```
       :JU  FB125      SYNCHRON handling block call
 NAME  :SYNCHRON
 SSNR  :    KY0,4     SSNR = 4, store in low byte
 SSDB  :    DB14      Allocate DB 14 as queue
 ANZW  :    KY0,6     Store condition codeword in flag word 6
 PAFE  :    F 1.0     Indicate para. assignment error in F 1.0
       :
       :AN  F 1.0     Is there no parameter assignment error?
       :JC  =F001     Yes, continue with FB 126
       :STP           No, then stop and abort (note 1)
       :
 F001  :JU  FB126     ACTIVE handling block call
 NAME  :ACTIVE
 SSDB  :    DB14      Processing the queue in DB 14
       :
       :A   F 6.6     SYNCHRON still in queue?
       :              (i.e. in cond. codeword FW 6 bit 14 = 0?)
       :JC  =F001     Yes, then return to FB 126 (note 2)
       :
       :AN  F 6.4     SYNCHRON processed without errors?
       :              (i.e. in cond. codeword FW 6 bit 12 = 0?)
       :BEC           Yes, then complete start-up FB
       :JU  PB100     No, go to error processing in PB 100
       :BE
```

Note 1: if this parameter assignment error bit (F P01.1) is set then the cold restart is aborted since the SSDB either does not exist or is not long enough.

Note 2: this loop forces the immediate and complete processing of the SYNCHRON function. Following this, by evaluating the ANZW bit 12 it is possible to determine whether this interface exists and has been correctly initialized. In the same way ANZW bit 14 indicates whether the SYNCHRON has been deleted from the queue again on completion of the initialization.

It should also be noted that the synchronization procedure can take up to 20 seconds per interface.

5.2 Data transfer with DIRECT functions (A-NR > 0)

After CP interface 4 has been synchronized, jobs can be triggered on the CP if they have already been programmed (these jobs are stored with programming "tools" for the specific CP, e.g. the CP 526 is programmed with the COM 526). These jobs are identified by their job numbers. Some of these jobs can also be linked with a data transfer. The triggering and execution of a data transfer for such jobs can be initiated by calling a SEND-DIRECT or RECEIVE-DIRECT. These calls are often made to be dependent on event flags of the process to be controlled or monitored.

Example 2: SEND-DIRECT with direct parameter assignment (DB)

Taking example 1 with interface number 4 and queue block DB 14, there is a job on the CP with job number 32 which involves sending data words from the CPU to the CP. Depending on the event flag F 2.3, 20 data words from data block 15 starting at data word 2 are to be transferred to the CP using direct parameter assignment. Data word 0 in DB 15 is to be used as the condition codeword and flag bit F 3.0 as the overflow bit.

Program section in OB 1:

```
      :
      :C   DB15
      :A   F 2.3      Event flag set? And
      :AN  D 0.14     SEND call not in queue (ANZW bit 14?)
      :JC  FB120      Yes, then call SEND-DIRECT block
NAME  :SEND
SSDB  :    DB14       Enter call in queue DB 14
A-NR  :    KY0,32     Call is for CP job number 32
ANZW  :    KY15,0     Store ANZW in DW 0 of DB 15
QTYP  :    KCDB       Direct param. assign. with data transfer
DBNR  :    KY0,15     Data to be sent stored in DB 15
QANF  :    KF+2       Beginning of data block from data word 2
QLAE  :    KF+20      Number of data words to be sent = 20
UELA  :    F 3.0      Overflow bit for the SEND call: F 3.0
      :A   F 3.0      Queue overflow owing to SEND?
      :JC  PB100      Yes, go to error processing in PB 100
      :
      :JU  FB126      ACTIVE handling block call
NAME  :ACTIVE
SSDB  :    DB14       Processing the queue in DB 14
      :AN  D 0.14     SEND deleted from queue?
      :JU  PB101      Yes, go to cond. code evaluation in PB 101
      :
```

Program in PB 101:

```
      :0   D 0.12     SEND complete with error?
      :0   D 0.13     SEND could not be processed?
      :JC  PB100      Yes, go to error processing in PB 100
      :BE
```

The SEND call is only entered in the queue once and the entry checked. The error-free execution of the SEND function is also checked.

Example 3: RECEIVE-DIRECT with indirect parameter assignment (XX)

Taking the second example with interface number 4 and queue DB 14 there is a further job on the CP with the job number 37 which is to transfer data from the CP to the CPU. In this data transfer the CPU is to receive 50 data words and store them in data block 16 from data word 13 onwards, using indirect parameter assignment. These destination parameters are stored in data block 17 from data word 5 onwards. The condition codeword for this job is flag word 8 and flag bit F 3.1 is the overflow bit.

Program section in OB 1:

```
      :
      :AN  F 8.6      RECEIVE call not in queue?
      :JC  PB20       Yes, then call RECEIVE in PB 20
      :
      :
      :JU  FB126      ACTIVE. block call
NAME  :ACTIVE
SSDB  :    DB14       Processing the queue in DB 14
      :AN  F 8.7      RECEIVE entry processed? And
      :AN  F 8.4      RECEIVE complete with errors?
      :S   F 4.1      Yes, then set flag 4.1 for error-free
      :               processing of the RECEIVE entry
      :
```

Program in PB 20:

```
      :JU  FB121      RECEIVE-DIRECT call
NAME  :RECEIVE
SSDB  :    DB14       Enter call in queue DB 14
A-NR  :    KY0,37     Call is for CP job number 37
ANZW  :    KY0,8      Store ANZW in FW 8
ZTYP  :    KCXX       Indirect param. assign. with data transfer
DBNR  :    KY0,17     Actual destination parameters are in DB 17
ZANF  :    KF+5       Start address of dest. parameters from DW 5
ZLAE  :    KF+0       irrelevant, must however be entered
UELA  :    F 3.1      Overflow bit for the RECEIVE call: F 3.1
      :A   F 3.1      Queue overflow due to RECEIVE call?
      :JC  PB100      Yes, go to error processing in PB 100
      :BE
```

Extract from DB 17:

```
      :
    4 :
    5 :   KS= DB    ┐                A block of data is to be
    6 :   KY= 0,16  > Actual         read into a DB of the PC, a
    7 :   KF= +13   | destination    total of 50 data words into
    8 :   KF= +50   ┘ parameters     DB 16 from data word 13.
    9 :
      :
```

If, in the above example, **SS** is specified as ZTYP instead of XX, then the transfer of the 50 DW's can only be executed later by means of a RECEIVE-ALL.

5.3 Background communication with ALL functions (A-NR = 0)

There are CP's which can have later data transfers written into their programs, with statements declaring the data blocks, the start addresses and the number of data words involved in the data transfer. This means that the CP is informed in advance of the source and destination parameters.

In the same way, while the user program is running SEND/RECEIVE calls of the QTYP/ZTYP = SS type can be processed with the data transfer not being carried out immediately. With calls of this type (SS) the CP can be informed of the source and destination parameters for the required data transfer.

The third possibility involves FETCH calls being processed in the user program for which the CP has already prepared the required data for transfer to the CPU. In this case the CP is once again already informed of the destination parameters.

Since, for the reasons mentioned above, data transfer requests can collect on the CP while the PC program is running, it is advisable to have handling blocks called from time to time in the program in order to trigger the background communication, with the CP providing the source and destination parameters for the communication. The handling blocks required for this are the **SEND-ALL** and **RECEIVE-ALL** functions.

When either of these functions is called relevant values are only specified on the block for the parameters SSDB, A-NR and ANZW, the ALL function being specified by A-NR = 0. Specifications made for the source and destination parameters will be ignored although they must be made.

When calling ALL functions, suitable scanning should ensure that the ALL function is only entered in the queue once.

This background communication is achieved by having ALL functions entered in the queue as required at various points in the user program, e.g.:

- in a time-controlled program section (OB 13) so that an ALL function is entered in the queue again at the most 100 ms following its processing;

- in the cyclic program section (OB 1 or FB 0) so that an ALL function is re-entered in the queue as soon as possible following its processing;

- whenever other blocks are not being called owing to process conditions, so that the load on the cycle is spread, or the process control has priority over the communication.

Example 4: background communication with SEND/RECEIVE-ALL

In the CP program with interface 4 the following is programmed as background communication in addition to jobs 32 and 37:

- when a SEND-ALL is processed in the CPU a floating-point number stored in double word 9 of data block 18 is to be transferred to the CP.

The SEND-ALL to be programmed in the CPU is to have flag word 12 as its condition codeword. The flag bit F 3.2 is to be used as the overflow bit, the SEND-ALL will be called in OB 13.

Program section in OB 1:

```
      :
      :JU  FB126
NAME  :ACTIVE
SSDB  :    DB14
```

Program section in OB 13:

```
      :
      :AN  F 12.6    SEND-ALL call not in queue?
      :JC  FB120     Yes, then call SEND-ALL
NAME  :SEND
SSDB  :    DB14
A-NR  :    KY0,0      Job number 0 for SEND-ALL
ANZW  :    KY0,12   _ ANZW in flag word 12
QTYP  :    KCxy     |
DBNR  :    KY0,0    |
QANF  :    KF+0     > irrelevant for ALL calls
QLAE  :    KF+0     _|
PAFE  :    F 3.2      Indicate param. assign. error in F 3.2
      :AN  F 3.2      Queue overflow owing to SEND-ALL?
      :JC  PB100      Yes, go to error processing in PB 100
      :
```

Extract from DB 18:

```
      :
    8 :
    9 :    KG = +1234000+02;      any KG number e.g. 12.34
   11 :
      :
```

A condition code evaluation is not necessary in this example as there is no ANZW-CP.

In order to avoid nesting the ACTIVE, the ACTIVE for processing the SEND-ALL is always called in OB 1 (see section 12.3).

5.4 Run times

The following tables show the run times required for entering SEND, RECEIVE, FETCH, RESET and SYNCHRON in the queue and the times required by CONTROL and ACTIVE when they are processed. The times shown have been rounded up or down to one decimal place and can be used to estimate the total run time of program sections.

Run times in ms for queue entries:

Handling block	when UELA = 1 or PAFE = 1	ANZW in flag word	ANZW in data word
SEND, RECEIVE FETCH, RESET	0.5 ms	3.1 ms	3.3 ms
SYNCHRON	0.5 ms	3.6 ms	3.8 ms

Run times in ms for processing the CONTROL:

Handling block	when PAFE = 1	ANZW in flag word	ANZW in data word
CONTROL-DIRECT	0.8 ms	2.4 ms	2.6 ms
CONTROL-ALL	–	1.7 ms	1.9 ms

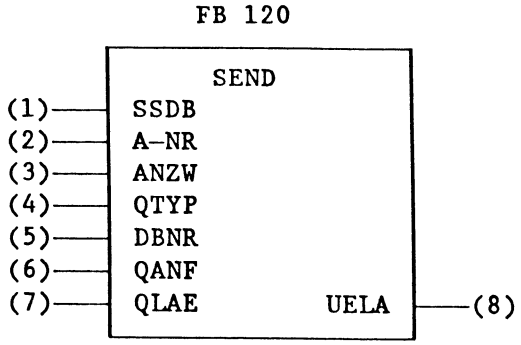
Run times in ms for processing the ACTIVE:

Handling block	Queue empty	Prep. time for data transfer	Transfer of 1 DW	Transfer of 128 DWs
ACTIVE	0.4 ms	20 ms *	20 ms	27 ms

* In total this time will be required 2 to 3 times to process a handling function depending on the ACTIVE calls.

6 Handling block FB 120: SEND

6.1 Block diagram



6.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSDB	Interface data block for queue in which SEND job is entered	B	
A-NR	Job no: number contained in low byte perm.: 0...255, **0** All function perm.: 0...255, 1...223 DIRECT function only the low byte is evaluated	D	KY
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word high byte > 0: high byte = no. of DB	D	KY
QTYP	Source type: depending on DB, XX, SS, NN the DBNR, QANF, QLAE have different meanings. Perm.: DB, XX, SS, NN	D	KC
DBNR	Data block no.: if QTYP = DB: DB contains data to be transferred If QTYP = XX,SS: source parameters perm.: 0...255, 3...255 only the low byte is evaluated	D	KY
QANF	Source data start: address of 1st DW or 1st source parameter in DBNR block perm.: +0...+255	D	KF
QLAE	Source data length: no. of DW's to be transferred from DBNR data block perm.: +1...+256	D	KF
UELA	Overflow bit: queue full, job cannot be entered. perm.: I, Q, F 0...255 . 0...7	Q	BI

Flags occupied	FW 202, FB 206
Data blocks called	SSDB, DB for ANZW
Function blocks called	none
Nesting depth	zero

6.3 SEND, functional description

The SEND block is used for the transfer of data or parameters from the central processing unit CPU of the PC to the interface on the CP. The CPU sends data to the interface (**send**). This block has two modes of operation:

- SEND-DIRECT (if the low byte of A-NR > 0)
- SEND-ALL (if the low byte of A-NR = 0)

6.3.1 SEND-DIRECT

The SEND-DIRECT function (**A-NR > 0**) is used to trigger the SEND job, specified by the job number A-NR, directly. The usual job numbers are 1 to 199. This function can, for example, be used with particular process statuses (which the user program in the CPU recognises and evaluates) to output a message on a printer.

The SEND-DIRECT function is only carried out by the handling block if there is no job running on the interface with the same job number; in this case bit 1 = 0 in the corresponding condition codeword. Otherwise the handling block will only go through an "idling run" in which only the condition codeword is updated.

For the SEND-DIRECT function the block initially only needs SSDB, A-NR, ANZW and UELA to be specified. Depending on the QTYP specified (DB, XX, SS, NN) the block triggers different functions.

- If **DB** is specified as the QTYP, the following parameters DBNR, QANF, QLAE are direct parameters. When the block is processed with these parameters (by the ACTIVE) the block of data identified by the parameters can be transferred directly from the CPU to the interface by the block.

- If **XX** is specified as the QTYP, the DBNR and QANF are indirect parameters (QLAE is then irrelevant). When the block is called they specify the DB and start address of the 4 actual source parameters. These actual parameters identify the block of data to be transferred and the data transfer from the CPU to the interface is triggered.

- If **SS** is specified as the QTYP then just as with type XX the DBNR and QANF are used to read out the 4 actual source parameters which are then transferred to the CP along with the job number. This makes it possible for the CP to supply these 4 source parameters at a later point when a SEND-ALL job is being processed (with A-NR = 0). Using the corresponding job number the block of data will then be transferred from the CPU to the interface.

- If **NN** is specified as the QTYP the parameters DBNR, QANF, QLAE in the block are then irrevelant. The CP supplies the handling block with the actual source parameters corresponding to the job number on the block A-NR > 0. With these parameters the handling block then triggers the data transfer from the CPU to the interface and updates the condition codeword belonging to this job number.

6.3.2 SEND-ALL

The SEND-ALL function (**A-NR = 0**) checks whether the interface has a SEND communication request for the CPU. If it has, the interface informs the handling block of the job number A-NR, the condition codeword ANZW-CP and the 4 source parameters; i.e. the interface determines which data are to be sent from the CPU and for which job this data transmission is intended. All the job numbers available on the CP can be used.

The SEND-ALL function requires the parameters SSDB, A-NR, ANZW and UELA to be specified in the block, whereby A-NR must be zero; the other source parameters specified in the block are irrelevant. The block receives a second condition codeword ANZW-CP and the actual source parameters from the interface.

Such background communication using SEND-ALL is, for example, a simple means of having process displays (process data) displayed on a monitor and regularly updated (cf. CP 526). The central processing unit neither needs to know the selected display number (= job number) nor does it need to refresh the data since the CP indicates the communication request in good time and also has the data required for the display sent by the next SEND-ALL to be called up.

A special case with the SEND-ALL function is the "remaining length transfer". This function is used by the CP 524 and CP 525 when operating a data terminal (DS 075) or a printer. In these special cases the data block to be transferred must be 261 words long (five words for the block header and DW 0 to DW 255).

6.4 SEND, operational sequence

The SEND function block is entered in the queue when it is called in the user program and bit 14 is set in the ANZW. When the queue is processed (by calling the ACTIVE block) this entry leads to the triggering of a job on an intelligent interface module with or without data transfer.

Initially the ANZW address ANZW-AG preset on the block is used. Under certain circumstances the CP also supplies a second ANZW address ANZW-CP. In this case the ANZW-CP will then be used, bits 14 and 15 are set, bits 12 and 13 cleared. If, following ACTIVE calls the job is completely processed, it is deleted from the queue (bits 14 and 15 are cleared, bit 12 or 13 may be set) and the ANZW-CP is duplicated in the ANZW-AG. The modification of the 4 bits 12-15 to their final status and the duplication of the ANZW takes place within one ACTIVE run.

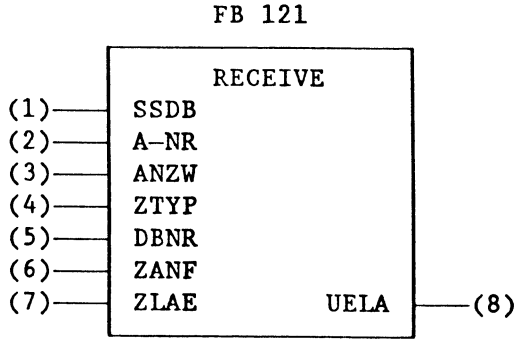
Bit 15 = 1 in the ANZW always indicates that the processing of this job has begun and that a data transfer may have already been partly carried out. The job is in a "critical" phase. Changing the data to be sent at this time could lead to "old" and "new" data being mixed, and should therefore be avoided.

The identical handling function (same block with same parameter specifications) should only be called again when the old job is complete and is no longer entered in the queue, i.e. when bit 14 = 0 in the ANZW. If, e.g. the SEND-ALL block is called cyclically in the user program, however, only on condition that in the ANZW bit 14 = 0, then it cannot be entered in the queue more than once; it requires little space and will nevertheless be processed regularly.

If with certain job numbers the transfer of the block of data has been inhibited by the user setting bit 7 = 1 in the condition codeword, then the function block should only be called dependent on this identifier.

7 Handling block FB 121: RECEIVE

7.1 Block diagram



7.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSDB	Interface data block for queue in which RECEIVE job is entered	B	
A-NR	Job no: number contained in low byte perm.: 0...255, **0** All function perm.: 0...255, 1...223 DIRECT function only the low byte is evaluated	D	KY
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word high byte > 0: high byte = no. of DB	D	KY
ZTYP	Dest. type: depending on DB, XX, SS, NN the DBNR, ZANF, ZLAE have different meanings. Perm.: DB, XX, SS, NN	D	KC
DBNR	Data block no.: if ZTYP = DB: DB contains area for data to be stored If ZTYP = XX,SS: dest. parameters perm.: 0...255, 3...255 only the low byte is evaluated	D	KY
ZANF	Dest. data start: address of 1st DW or 1st dest. parameter in DBNR block perm.: +0...+255	D	KF
ZLAE	Dest. data length: no. of DW's to be transferred to DBNR data block perm.: +1...+256	D	KF
UELA	Overflow bit: queue full, job cannot be entered. perm.: I, Z, F 0...255 . 0...7	Z	BI

Flags occupied	FW 202, FB 206
Data blocks called	SSDB, DB for ANZW
Function blocks called	none
Nesting depth	zero

7.3 RECEIVE, functional description

The RECEIVE block is used for the transfer of data to the central processing unit CPU of the PC from the interface on the CP. The CPU receives data. This block has two modes of operation:

- RECEIVE-DIRECT (if the low byte of A-NR > 0)
- RECEIVE-ALL (if the low byte of A-NR = 0)

7.3.1 RECEIVE-DIRECT

The RECEIVE-DIRECT function **(A-NR > 0)** is used to trigger the RECEIVE job, specified by the job number A-NR, directly. The usual job numbers are 1 to 199. This function can, for example, be used with particular process statuses (which the user program in the CPU recognises and evaluates) to receive measured values from various control loops.

The RECEIVE-DIRECT function is only carried out by the handling block if there are data available on the interface of the CP (bit 0 = 1 in the ANZW). Otherwise the handling block will only go through an "idling run" in which only the condition codeword is updated.

For the RECEIVE-DIRECT function the block initially only needs SSDB, A-NR, ANZW and UELA to be specified. The block only requires destination parameters if a handshake procedure is started (bit 0 = 1 in ANZW). Depending on the ZTYP specified (DB, XX, SS, NN) the block triggers different functions:

- If **DB** is specified as the ZTYP, the following parameters DBNR, ZANF, ZLAE are direct parameters. When the block is processed with these parameters (by the ACTIVE) the block of data identified by the parameters can be transferred directly from the interface to the CPU by the block.

- If **XX** is specified as the ZTYP, the DBNR and ZANF are indirect parameters (ZLAE is then irrelevant). When the block is called they specify the DB and start address of the 4 actual destination parameters. These actual parameters identify the data area to which the data will be transferred.

- If **SS** is specified as the ZTYP then, just as with type XX, the DBNR and ZANF are used to read out the 4 actual destination parameters which are then transferred to the CP along with the job number without a data transfer taking place. This makes it possible for the CP to supply these 4 destination parameters and A-NR at a later point when a RECEIVE-ALL job is being processed thus defining the memory area in the CPU to which the data will be transferred from the interface.

- If **NN** is specified as the ZTYP the parameters DBNR, ZANF, ZLAE in the block are then irrevelant. The CP supplies the handling block with the actual destination parameters corresponding to the job number in the block A-NR > 0. With these parameters the handling block then triggers the data transfer from the interface to the CPU and updates the condition codeword belonging to this job number.

7.3.2 RECEIVE-ALL

The RECEIVE-ALL function (**A-NR = 0**) checks whether the interface has a RECEIVE communication request. If it has, the interface makes the destination parameters available; i.e. the interface determines where the data to be transferred to the CPU by the handling block are to be stored and for which job and job number these data are intended. All the job numbers used can occur. When programming PC's and CP's it should not be forgotten that ALL functions change a variety of memory areas in the programmable controller.

The RECEIVE-ALL function requires the parameters SSDB, A-NR, ANZW and UELA to be specified. The other destination parameters specified in the block are irrevelant. The block receives the address of a second condition codeword ANZW-CP and the actual destination parameters from the interface.

Such background communication is, for example, a simple means of having input values (entered at a terminal keyboard) transferred to the CPU. The central processing unit neither needs to know the selected display number (= job number) nor does it need to know whether and when input values are available, since the CP indicates the communication request and the next RECEIVE-ALL transfers the values to the CPU.

Data which must first be fetched by the CP via a bus coupling from other partners or must be generated in some other way are transferred to the CPU by a combination of the FETCH and the RECEIVE-ALL functions (see block description).

A special case with the RECEIVE-ALL function is the "remaining length transfer". This function is used by the CP 524 and CP 525 when operating a data terminal (DS 075) or a printer. In these special cases the data block to be transferred must be 261 words long (five words for the block header and DW 0 to DW 255).

7.4 RECEIVE, operational sequence

The RECEIVE function block is entered in the queue when it is called in the user program and bit 14 is set in the ANZW. When the queue is processed (by calling the ACTIVE block) this entry leads to the triggering of a job on an intelligent interface module with or without data transfer.

Initially the ANZW address preset in the block is used. Under certain circumstances the CP also supplies a second ANZW address ANZW-CP. In this case the ANZW-CP will then be used, bits 14 and 15 are set, bits 12 and 13 cleared. If, following ACTIVE calls the job is completely processed, it is deleted from the queue (bits 14 and 15 are cleared, bit 12 or 13 may be set) and the ANZW-CP is duplicated in the ANZW-AG. The modification of the 4 bits 12-15 to their final status and the duplication of the ANZW takes place within one ACTIVE run.

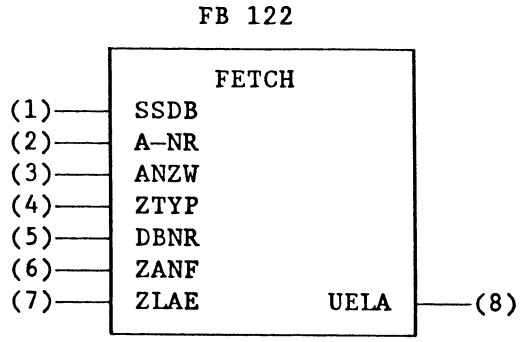
Bit 15 = 1 in the ANZW always indicates that the processing of this job has begun and that a handshake has started. The job is in a "critical" phase. Changing the data to be received at this time could lead to "old" and "new" data being mixed, and should therefore be avoided.

The identical handling function (same block with same parameter specifications) should only be called again when the old job is complete and is no longer entered in the queue, i.e. when bit 14 = 0 in the ANZW. If, e.g. the RECEIVE-ALL block is called cyclically in the user program, however, only on condition that in the ANZW bit 14 = 0, then it cannot be entered in the queue more than once; it requires little space and will nevertheless be processed regularly.

If with certain job numbers the transfer of the block of data has been inhibited by the user setting bit 7 = 1 in the condition codeword, then the function block should only be called dependent on this identifier.

8 Handling block FB 122: FETCH

8.1 Block diagram



8.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSDB	Interface data block for queue in which FETCH job is entered	B	
A-NR	Job no: number contained in low byte perm.: 0...255, 1...223 only the low byte is evaluated	D	KY
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word high byte > 0: high byte = no. of DB	D	KY
ZTYP	Dest. type: depending on DB, XX, SS, NN the DBNR, QANF, QLAE have different meanings. Perm.: DB, XX, SS, NN	D	KC
DBNR	Data block no.: if ZTYP = DB: DB contains area for data to be stored If QTYP = XX,SS: dest. parameters perm.: 0...255, 3...255 only the low byte is evaluated	D	KY
ZANF	Dest. data start: address of 1st DW or 1st dest. parameter in DBNR block perm.: +0...+255	D	KF
ZLAE	Dest. data length: no. of DW's to be transferred to DBNR data block perm.: +1...+256	D	KF
UELA	Overflow bit: queue full, job cannot be entered. perm.: I, Q, F 0...255 . 0...7	Q	BI

Flags occupied	FW 202, FB 206
Data blocks called	SSDB, DB for ANZW
Function blocks called	none
Nesting depth	zero

8.3 FETCH, functional description

The FETCH block triggers a job on the CP to fetch data. This gives the CPU access to data which are not present on the CP but which must first be generated or acquired by the CP, e.g. via the SINEC H1 bus from a different automation system.

The FETCH block only uses the mode: **FETCH-DIRECT**.

By transferring the job number (typically A-NR = 1 to 199), the destination parameters and the address of the condition codeword, the FETCH block informs the interface which data are required, where these data are to be stored in the CPU (destination parameters) and which condition codeword is to be updated. As soon as the requested data are available on the CP the interface provides the parameters for the data transfer; the FETCH block itself does not carry out the data transfer.

The final data transfer is triggered by the RECEIVE-ALL job; the ANZW-AG of the FETCH job becomes the ANZW-CP of the RECEIVE job.

For the FETCH-DIRECT function the block initially requires the parameters SSDB, A-NR, ANZW and UELA. Depending on the ZTYP specification (DB, XX, SS, NN) the block then triggers different functions.

- If **DB** is specified as the ZTYP then the following parameters DBNR, ZANF, ZLAE are direct parameters. With these parameters when the block is called in the queue it can have the CP prepare the data to be transmitted and the destination parameters on the interface where they can be called later by a RECEIVE-ALL.

- If **XX** is specified as the ZTYP, DBNR and ZANF are indirect parameters (ZLAE is then irrelevant). When the block is called they specify the DB and the start address of the 4 actual destination parameters. The CP then makes the fetched data and the 4 actual destination parameters available on the interface for a subsequent RECEIVE-ALL job.

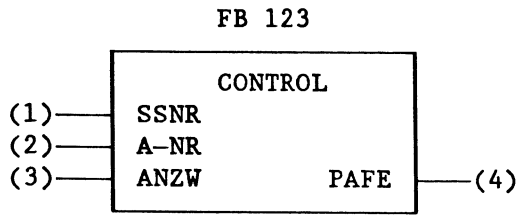
- If **NN** is selected as the ZTYP the parameters DBNR, ZANF and ZLAE are irrelevant. The CP only uses the job number A-NR specified in the block to make the fetched data and the actual destination parameters available on the interface for a subsequent RECEIVE-ALL job.

8.4 FETCH, operational sequence

The FETCH call leads to an entry being made in the queue. When the queue is being processed by means of ACTIVE calls the entry is initiated and the parameters are transferred to the CP. The actual data transfer is brought about by a later RECEIVE-ALL.

9 Handling block FB 123: CONTROL

9.1 Block diagram



9.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSNR	Interface number: number assigned by jumper setting on CP perm.: 0...255, 0...255 only low byte (with no.) is evaluated	D	KY
A-NR	Job no: number contained in low byte perm.: 0...255 **0** ALL function perm.: 0...255, 1...223 DIRECT function only low byte (with no.) is evaluated	D	KY
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word high byte > 0: high byte = no. of DB	D	KY
PAFE	Parameter assignment error bit: error display for job with job number A-NR perm.: I, Q, F 0...255 . 0...7	Q	BI

Flags occupied	FW 202, FW 204, FB 206
Data blocks called	DB for ANZW
Function blocks called	none
Nesting depth	zero

9.3 CONTROL, functional description

The CONTROL block is used to scan the status information on the SSNR interface. The block uses the two modes:

- **CONTROL-DIRECT** (when the low byte of A-NR > 0)
- **CONTROL-ALL** (when the low byte of A-NR = 0)

9.3.1 CONTROL-DIRECT

If the function block CONTROL is assigned a job number > 0, when it is called, it updates bits 0 to 3 and bits 8 to 11 of the condition codeword; i.e. the job status of the job with the specified job number is transferred to the ANZW-AG, bits 4 to 7 and 12 to 15 remaining unchanged.

9.3.2 CONTROL-ALL

If the CONTROL block is assigned the job number 0 it stores the job number of the job currently being processed by the CP (i.e. by the interface) in the low byte of the condition codeword ANZW-AG. The high byte of the ANZW-AG remains unchanged.

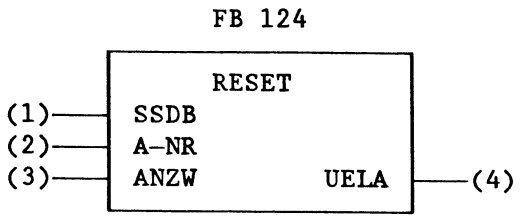
9.4 CONTROL, operational sequence

When the CONTROL handling block is called in the user program it is executed immediately, i.e. without being entered in the queue.

The output parameter PAFE indicates whether or not a parameter assignment error has been made, e.g. specification of an illegal interface number.

10 Handling block FB 124: RESET

10.1 Block diagram



10.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSDB	Interface data block for queue in which RESET job is entered	B	
A-NR	Job no: number contained in low byte perm.: 0...255, **0** All function perm.: 0...255, 1...223 DIRECT function only the low byte is evaluated	D	KY
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word high byte > 0: high byte = no. of DB	D	KY
UELA	Overflow bit: queue full, job cannot be entered. perm.: I, Q, F 0...255 . 0...7	Q	BI

Flags occupied	FW 202, FB 206
Data blocks called	SSDB, DB for ANZW
Function blocks called	none
Nesting depth	zero

10.3 RESET, functional description

The RESET block is used to clear or abort one or all of the jobs, initiated by the CPU and currently running on the CP. The block can use two modes:

- **RESET-DIRECT** (if A-NR > 0)
- **RESET-ALL** (if A-NR = 0)

10.3.1 RESET-DIRECT

If the RESET block is assigned a job number **A-NR > 0** the job on the CP will be aborted and brought to a defined inactive status when the RESET is processed.

10.3.2 RESET-ALL

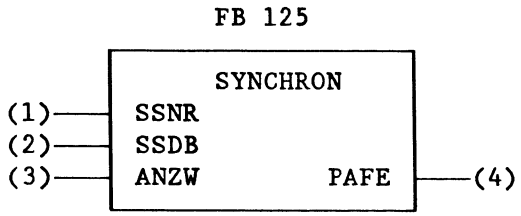
If the RESET block is assigned the job number **A-NR = 0** then when it is called all the currently running jobs on the corresponding CP interface will be reset and aborted.

10.4 RESET, operational sequence

When it is called in the user program the handling block RESET is entered in the queue of the specified interface. When the queue is processed this entry causes one or all of the jobs currently running on this interface to be aborted.

11 Handling block FB 125: SYNCHRON

11.1 Block diagram



11.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSNR	Interface number: number assigned by jumper setting on CP perm.: 0...255, 0...255 only low byte (with no.)is evaluated	D	KY
SSDB	Interface data block for queue belonging to SSNR	B	
ANZW	Cond. codeword: low byte = no. of DW/FW perm.: **0** , 0...198 for flag word perm.: 3...255, 0...255 for data word Bits 0 – 11 are irrelevant, only bits 12–15 (sequence management) are used	D	KY
PAFE	Parameter assignment error bit: error display for job with job number A–NR perm.: I, Q, F 0...255 . 0...7	Q	BI

Flags occupied	FW 200, FW 202, FW 206
Data blocks called	SSDB, DB for ANZW
Function blocks called	none
Nesting depth	zero

11.3 SYNCHRON, functional description

The SYNCHRON block is used to initialize the interface SSNR of a CP and establishes the allocation of the queue block SSDB to the interface SSNR. At the same time the working area of the queue block SSDB is set up and checked to establish whether the data block is long enough for the handling block entries. In addition to this, the dual-port RAM of the interface is preassigned.

11.4 SYNCHRON, operational sequence

SYNCHRON runs in two stages. In the first stage the queue block is set up and in the second stage the dual-port RAM is preassigned.

When it is called, SYNCHRON sets up the data block SSDB as the queue block which has the interface SSNR allocated to it. This initializes the queue i.e. its working area is deleted and set up again and SYNCHRON enters itself as the first call in the queue. Following this, bit 14 of the ANZW is set to 1 and all other bits cleared.

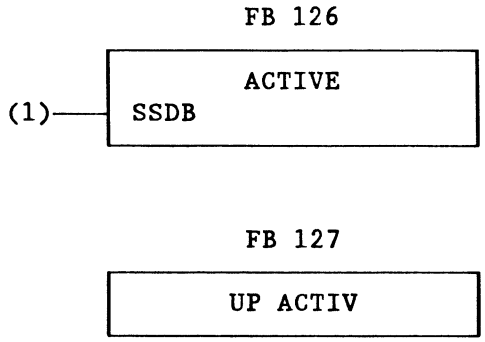
This entry is processed when an ACTIVE block is called; the dual-port RAM is cleared and preassigned. With this the SYNCHRON is complete and is deleted from the queue again and the ANZW bit 14 is reset.

If, however, there is no SSDB present or it is not long enough, the block is exited and the PAFE bit set; ANZW is not changed and the SYNCHRON cannot be entered in the queue since this cannot be initialized.

If this occurs the user should stop the programmable controller since SEND, RECEIVE, FETCH and RESET jobs require a correctly initialized queue. If the programmable controller is not stopped the processing of these jobs could lead to errors.

12 Handling blocks FB 126: ACTIVE
FB 127: UP ACTIV

12.1 Block diagrams



12.2 Description of parameters

Par.	Meaning and permissible ranges	Param type	Data type
SSDB	Interface data block for queue, which is processed by the ACTIVE block	B	

Occupied flags FB 200 to FB 255
Occupied system data BS 254

Data blocks called
- DB for queue SSDB,
- DB for condition codeword ANZW-AG,
- DB for condition codeword ANZW-CP,
- DB for source and destination data with indirect parameter assignment
- DB for data sent and received

Function blocks called FB 127 UP ACTIV

Nesting depth one

12.3 ACTIVE, functional description

The ACTIVE function block is used for processing entries in the queue. ACTIVE calls the UP-ACTIV function block as a subroutine.

A time interrupt (OB 13) or process interrupt (OB 2) could be inserted at such a block boundary and processed. In order to avoid nesting the FB ACTIVE it must only be called either in the cyclic section (OB 1 or FB 0) or only in the time-controlled section (OB 13) of the user program.

If the FB ACTIVE is called in the cyclic section (OB 1 or FB 0) and the time or process interrupt-driven programs also use the flags from flag byte 200 onwards (standard FB's, other handling blocks) then the flags with multiple assignment should be saved at the beginning of the interrupt processing and rewritten when it is completed.

12.4 ACTIVE/UP ACTIV, operational sequence

When an ACTIVE is called in the user program it processes the first queue entry in the specified interface data block SSDB. The processing of a handling function in the queue normally requires 4 to 5 active calls. Once an entry has been fully processed, it is deleted from the queue. The entries are processed in the order in which they are entered in the queue.

The data exchange between the PC and CP during an ACTIVE run takes place in blocks of max. 256 bytes. If, e.g. 400 bytes (DW 0 to DW 199) are to be transferred, the first data block contains 256 bytes (DW 0 to DW 127), the second data block contains the remaining 144 bytes (DW 128 to DW 199), the data transfer therefore requires that ACTIVE is run through more than once.

13 List of keywords

Abbreviation	Keyword	Page
	ACTIVE	4, **47**
	ALL function	**12**, 14, **28**
	Background communication	**28**, 29
CPU	Central processing unit	2
	Cold restart	5
ANZW	Condition codeword	8, **13**, **18**
	CONTROL	4, 20, **41**
CP	Communications processor	2, **3**
DBNR	Data block number	8, **15**, 16
	Data destination	9, **14**
	Data management	19, 21, **22**
	Data source	9, **14**
	Destination parameter	9, **14**
ZLAE	Destination parameter length	8, **15**, 16
ZANF	Destination parameter start address	8, **15**, 16
	Destination type	8, **14**
	DIRECT function	**13**, **26**
	Direct parameter	9, **11**
	Dual-port RAM	2, 3, **46**
	Error number	19, 20, **22**
	FETCH	4, 20, **39**
	Handling block	**2**, **31**
	Handshake	6, **21**, 38
	High byte	12, 13
	Indirect parameter assignment	**9**, **10**, 11
	Initialization	5, **24**, 46
IP	Intelligent I/O's	2
	Interface	**3**, 5, 6
SSDB	Interface data block	**5**, 6, **12**
SSNR	Interface number	**3**, 8, **12**
	Job	3
	Job management	19, 20, **21**
A-NR	Job number	3, 8, **12**
	Job status	20
	Low byte	12, 13
	Multiprocessing system	7
	Output parameter	17
OB	Organization block	5, **24**
	Page	2
	Parameter	8
PAFE	Parameter assignment error	8, **17**, 18
	Parameter description	12
	Parameter designation	8
	Parameter field	10
PC	Programmable controller	2, 3
	Queue	4, **5**, **17**
UELA	Queue overflow	8, **17**, 18
	RECEIVE	4, 20, **35**
	Remaining length transfer	33, **37**
	RESET	4, 20, **43**
	Run time	30
	Sequence management	19, 21, **23**
	SEND	4, 20, **31**
	SIMATIC S5 bus	2, 7
	SINEC H1 bus	40

Abbreviation	Keyword	Page
	Source parameter	9, **14**
QLAE	Source parameter length	8, **15**, 16
QANF	Source parameter start address	8, **15**, 16
	Source type	8, **14**
	STATUS	18, **20**
	SYNCHRON	4, 25, **45**
	Synchronization	5, 24, **46**
	UP ACTIV	4, **47**

Siemens Aktiengesellschaft

C79000-B8576-C325-03
Order from: Gerätewerk Karlsruhe
Printed in the Federal Republic of Germany
AG 0889 0.3 SK 50 En