

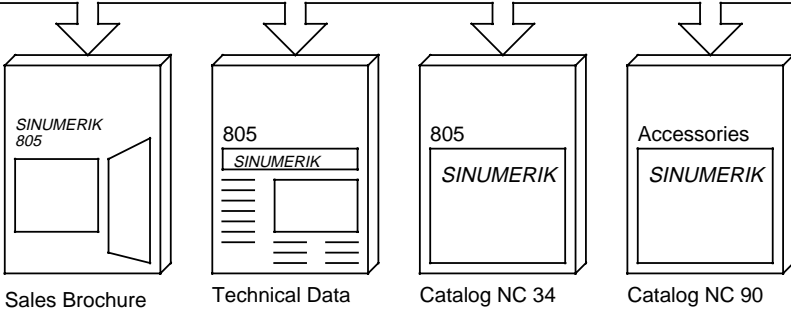
SINUMERIK 805
Software Version 4
PLC Programming

Planning Guide

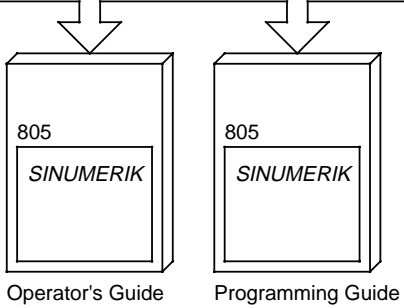
11/91 Edition

SINUMERIK 805

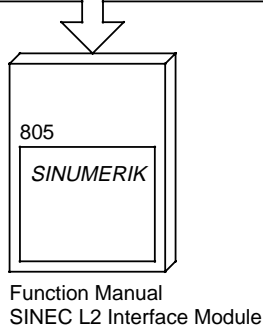
General Documentation



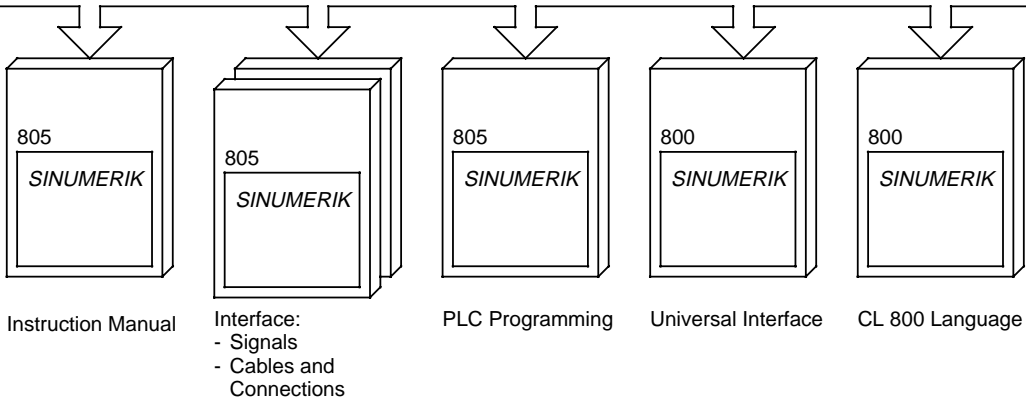
User Documentation



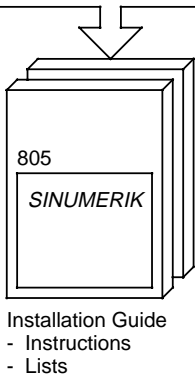
User/Manufacturer and Service Documentation



Manufacturer Documentation



Service Documentation



SINUMERIK 805

**Software Version 4
PLC Programming**

Planning Guide

Manufacturer Documentation

November 1991 Edition

SINUMERIK® documentation

Brief details of this edition and previous editions are listed below.

The status of each edition is shown by the code in the "Remarks" column.

Status code in "Remarks" column:

A . . . New documentation **B** . . . Unrevised reprint with new Order No.
C . . . Revised edition with new status. If factual changes have been made on a page since the last edition, this is indicated by a new edition coding in the header on that page.

Edition	Order No.	Remarks
01.90	6ZB5 410-0CM02-0BA0	A
10.90	6ZB5 410-0CM02-0BA1	C
01.91	6ZB5 410-0CM02-0AA2	C
11.91	6ZB5 410-0CM02-0AA3	C

Other functions not described in this documentation might be executable in the control. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

This publication was produced on the Siemens 5800 Office System.
Subject to change without prior notice.

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

© Siemens AG 1990 All Rights Reserved

Introduction

1

Organization, Program and Sequence Blocks

2

Data Blocks

3

Function Blocks

4

Program Organization

5

Integral Function Blocks

6

Transfer Parameters and Operating System Machine

7

Error Analysis

8

STEP 5 Command Set with Programming Examples

9

Rules of Compatibility between LAD, CSF and STL

10

Contents

	Page
1	Introductory Remarks 1-1
1.1	Application, memory capacity 1-1
1.2	STEP 5 programming language 1-2
1.3	Programming 1-3
1.3.1	Program structure 1-3
1.3.2	Program organization 1-4
1.3.3	Program scanning 1-5
1.3.4	Influencing factor: User program size 1-6
2	Organization, Program and Sequence Blocks 2-1
2.1	Programming program blocks 2-1
2.2	Calling program blocks 2-2
3	Data Blocks 3-1
3.1	Programming data blocks 3-1
3.2	Calling data blocks 3-2
3.3	Data blocks created by the operating system 3-4
4	Function Blocks 4-1
4.1	General remarks 4-1
4.2	Structure of function blocks 4-2
4.2.1	Block header 4-2
4.2.2	Block body 4-2
4.3	Calling and initializing function blocks 4-3
4.3.1	Call statement 4-3
4.3.2	Parameter list 4-3
4.4	Programming function blocks 4-4
4.4.1	Library number 4-4
4.4.2	Name of the function block 4-4
4.4.3	Formal operand (block parameter name) 4-5
4.4.4	Block parameter types 4-6
4.4.5	Block parameters and permitted actual parameters 4-7
5	Program Organization 5-1
5.1	General remarks 5-1
5.2	Cyclic scanning 5-2
5.2.1	Interrupt capability 5-2
5.2.2	Response time and cycle time 5-3
5.2.3	Programming the cyclic program 5-3
5.2.4	Interface between operating system and cyclic program 5-4
5.2.5	Basic program organization 5-5
5.3	Interrupt processing 5-7
5.3.1	Programming the interrupt service routine 5-7
5.3.2	Interface between operating system and the interrupt 5-7
5.3.3	Response time 5-8

6	Integral Function Blocks	6-1
6.1	General remarks	6-1
6.2	FB identifiers	6-2
6.3	Function macros	6-3
	FB 11 EINR-DB Generate data blocks	6-4
	FB 60 BLOCK-TR Block transfer	6-6
	FB 61 NCD-LESE Read NC data	
	FB 62 NCD-SCHR Write NC data	6-8
	FB 65 M STACK Transfer flags to stack	6-18
	FB 66 STACK M Flag stack to transfer flag area	6-18
7	Transfer Parameters and Operating System Machine Data	7-1
7.1	Transfer parameters	7-1
7.2	PLC machine data SINUMERIK 805	7-4
7.3	PLC machine data bits SINUMERIK 805	7-5
8	Error Analysis	8-1
8.1	Types of error	8-1
8.2	Interrupt stack	8-3
8.3	Detailed error code	8-5
8.3.1	Display on programmer	8-5
8.3.2	Display on NC screen	8-6
8.4	Alarm List	8-7
9	STEP 5 Command Set with Programming Examples	9-1
9.1	Memory organization	9-1
9.1.1	Changing the segment switch	9-3
9.1.2	Block lists	9-3
9.2	General notes	9-6
9.2.1	Numeric representation	9-6
9.2.2	Condition codes of the PLC	9-8
9.2.3	STEP 5 command representation	9-9
9.3	Basic operations	9-11
9.3.1	Logic operations, binary	9-11
9.3.2	Storage operations	9-14
9.3.3	Load and transfer operations	9-17
9.3.4	Timing and counting operations	9-19
9.3.5	Comparison operations	9-27
9.3.6	Block calls	9-31
9.3.7	Code operations	9-33
9.3.8	Arithmetic operations	9-34
9.3.9	Other operations	9-35
9.4	Supplementary operations (with function blocks only)	9-34
9.4.1	Logic operations, binary	9-35
9.4.2	Setting operations	9-35
9.4.3	Timing and counting operations	9-36
9.4.4	Enabling operations for timing and counting operations	9-38

9.4.5	Bit test operations (with function blocks only)	9-39
9.4.6	Coding, load and transfer operations	9-40
9.4.7	Logic operations, digital	9-41
9.4.8	Shift operations	9-41
9.4.9	Conversion operations	9-42
9.4.10	Decrementing / Incrementing	9-42
9.4.11	Jump operations	9-43
9.4.12	Processing operations	9-46
9.4.13	STEP 5 commands with direct memory access	9-48
9.4.14	Other operations	9-49
9.5	Command Set with Programming Examples	9-50
10	Rules of Compatibility between the LAD, CSF and STL Methods of Representation	10-1
10.1	General	10-1
10.2	Rules of compatibility for graphic program input (LAD, CSF)	10-2
10.3	Rules of compatibility for program input in a statement list	10-4

Preliminary Remarks

Notes for the reader

This manual is intended for the manufacturers of machine tools using SINUMERIK 805.

The Guide describes the program structure and the operation set of the PLC, and explains how basic PLC software and user programs, which comprise data blocks, function blocks, program blocks and organization blocks, are constructed.

The SINUMERIK documentation is organized in three levels:

- User documentation
- Manufacturer documentation and
- Service documentation

The **Manufacturer Documentation** for **SINUMERIK 805** is divided into the following:

- Instruction Manual
- Description of interfaces
 - Part 1: Signals
 - Part 2: Cables and hardware
- PLC Programming

Additional SINUMERIK publications are also available for all SINUMERIK controls (e. g. publications on the Universal Interface, Measuring Cycles, CL 800 Cycles language).

Please contact your Siemens regional office for further details.

Technical information

This documentation applies to software version 4

1 Introductory Remarks

1.1 Application, memory capacity

The PLC is a powerful interface controller. It has no inherent hardware, and is used as "software PLC" in the SINUMERIK 805.

The PLC is responsible for control of machine-related functional sequences such as

- Controlling auxiliary axes
- Controlling tool-changers
- Gathering the signals generated by the machine's monitoring devices.

Special features:

On the SINUMERIK 805, all NC and PLC jobs are executed by a common processor (Intel 80 186). A specially developed coprocessor (COP) is responsible for high-speed execution of binary operations and byte and word commands of the STEP 5 user program (e. g. A I 3.0, L FW3, etc.).

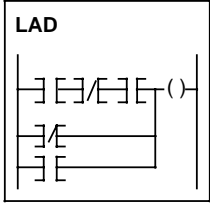
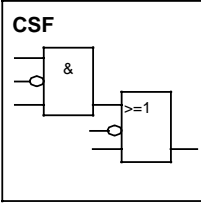
The single-processor structure calls for strict priority scheduling of NC and PLC jobs. NC functions such as position control and block preparation, but also the PLC "Interrupt processing" function (OB 2), have highest priority, while scanning of the cyclic program (OB 1) is of lesser importance. The amount of processor time available to the PLC has been restricted to a maximum of 20% in order to ensure that the NC can fulfill its primary objectives (positioning accuracy, short block change times) even in extreme situations, e. g. where an exceptionally long STEP 5 program or frequent interrupt servicing is involved. It is also possible to influence the manner in which the cyclic user program scanned via machine data (see Section 1.3.4).

PLC memory capacity of the SINUMERIK 805

User memory (OB, PB, FB, SB):	8 Kbytes
	16 Kbytes (from SW 4.1, Basic Version)
Data memory (DB):	4 Kbytes

1.2 STEP 5 programming language

The operations available in STEP 5 enable the user to program functions ranging from simple binary logic to complex digital functions and basic arithmetic operations. Depending on the programmer (PG) used a STEP 5 program may be written in the form of a control system flowchart (CSF), ladder diagram (LAD) or statement list (STL), thus enabling the programming method to be adapted to the application. The machine code (MC5) generated by the programmers is identical for all three. Depending on the programmer (PG) used, the user program can be translated from one method of representation to another by conforming to certain programming conventions (cf. Section10).

Ladder diagram	Statement list	Control system flowchart
Programming with symbols similar to those used in schematic circuit diagrams Complies with DIN 19239 (draft)	Programming with using mnemonics designating functions Complies with DIN 19239 (draft)	Programming with graphic symbols Complies with IEC 117 - 15 DIN 40700 DIN 40719 DIN 19239 (draft)
LAD 	STL <pre> A I AN I A I ON I O I = Q </pre>	CSF 

Methods of representing the STEP 5 programming language

1.3 Programming

1.3.1 Program structure

The PLC software comprises the operating system, the basic software and the user program. The operating system contains all statements and declarations for internal system functions. The basic software has a flexible interface to the operating basic functions (e. g. generation of data blocks, NC-PLC interface initialization, signal interchange with the peripherals). The basic software also contains pretested function blocks written in STEP 5 and assembled to form function macros.

The operating system and the basic software are integral components of the PLC; they are supplied on EPROMs together with the NC system program, and may not be modified in any way.

The user program is the total of all statements and declarations/data programmed by the user.

The PLC structure makes structured programming essential, i. e. the program must be divided into individual, self-contained sections called blocks. This method offers the following advantages:

- Easy, lucid programming, even of large programs
- Easy standardization of program sections
- Simple program organization
- Fast, easy modification
- Simple program testing
- Easy start-up

A number of block types, each of which is used for different tasks, is available for structuring the user program:

- Organization blocks (OBs)
The OBs serve as interface between operating system and user program.
- Program blocks (PBs)
The PBs are used to break the user program down into technologically oriented sections.
- Function blocks (FBs)
The FBs are used to program frequently recurring complex functions (such as individual controls, reporting, arithmetic and PID control functions).
- Sequence blocks (SBs)
SBs are special forms of program blocks used primarily for processing sequencers.
- Data blocks (DBs)
DBs are used for storing data or texts, and differ in both function and structure from all other block types.

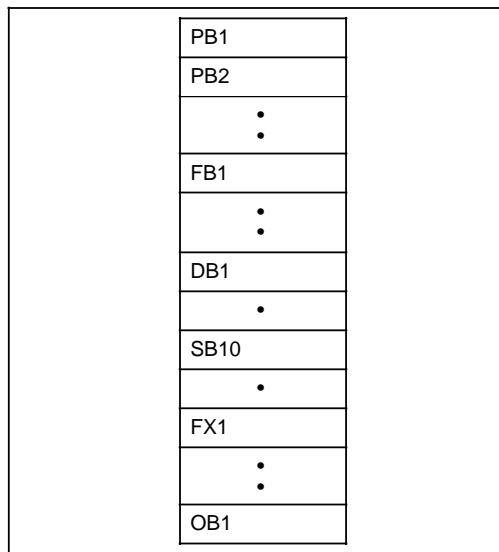
With the exception of the organization blocks, the maximum number of programmable blocks of each type is 255. The number of organization blocks may not exceed 64; of these, only OB 20, PB 1 and OB 2 are serviced by the operating system (cf. Section 5).

The programmer stores all programmed blocks in arbitrary order in program memory (Fig. 1.2).

1.3.2 Program organization

The manner in which the program is organized determines whether and in what order the program, function and sequence blocks are executed. The order in which these blocks are involved is stipulated by programming the relevant calls (conditional or unconditional) in organization blocks (see Section 5.4, "Cyclic program").

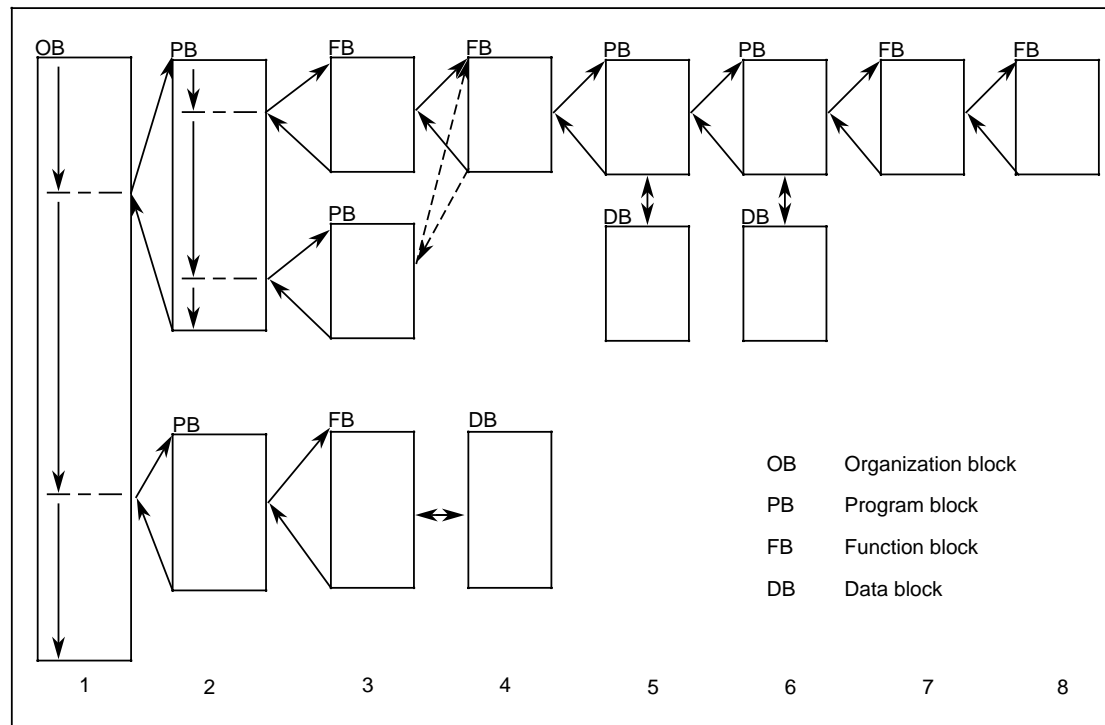
Like the other blocks, the organization blocks are stored in user memory.



Different organization blocks are provided for various methods of program execution (cf. Section 5.2).

Organization, program, function and sequence blocks can invoke other program, function and sequence blocks. The user program cannot call organization blocks. The maximum permissible nesting depth for organization blocks is 12, not including an accompanying data block, if any.

Storing the blocks in arbitrary order in program memory



Typical program organization in STEP 5 (nesting depth 8) (max. nesting depth 12)

1.3.3 Program scanning

The user program can be scanned in three ways:

- **One-shot scan**

One-shot scan in the controller's restart routine following "Power up" or "Hardware reset". The statements to be executed must be programmed in OB 20.

- **Cyclic scanning**

For cyclic scanning OB1 is available. This block is passed at intervals of 60ms and calls the programmed blocks.

- **Interrupt scan**

Event-driven, one-shot execution of the statements in OB 2, e. g. in response to an interrupt. Prerequisites for event-driven execution are as follows:

- Normal termination of the controller's restart routine
- Interrupt enable via PLC-MD 2002, bit 0="0".
- A change in the interrupt input byte specified per PLC-MD 0.

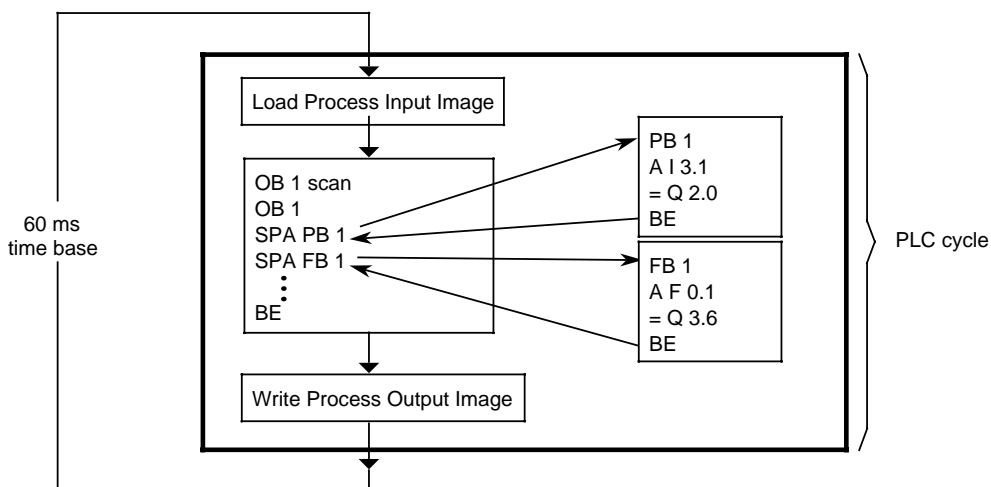
Cyclic scanning can be interrupted by the high-priority NC functions' interrupt service routine or delayed pending termination of OB 2 following execution of each STEP 5 command.

Execution of a PLC cycle; cyclic program scanning

Before OB 1 is called and started, the basic program loads the momentary state of all peripheral inputs into working memory. All input scans in the cycle which follows refer to this loaded status image. It is also known as "Process Input Image" or PII for short.

OB 1 is now called. This executes the blocks that have been called one after the other. This process can be interrupted by one or several interrupt scans (OB 2).

When OB 1 has been worked through completely, the system program transfers the statuses of the outputs (resulting from program execution) from the working memory to the peripheral outputs (MDP submodules, for example).



1.3.4 Influencing factor: User program size

As mentioned in Section 1.1, execution of the STEP 5 program should not take up more than 15 % of the total available CPU time, nor should the interrupt service routine in OB 2 take up more time than necessary because of its high priority, as this would delay execution of important NC functions.

For this purpose, the execution times of both the cyclic user program and the interrupt service routine are monitored at the hardware level. The permissible runtimes can be set in PLC MD 1 and 3, whereby confirmation of the defaults ensures that the "integrated PLC" will not take up more than the admissible amount of CPU time. When these values are exceeded, the PLC goes to the state specified in bit 0 or 1 of PLC MD 2003; when the defaults are used, the "integrated PLC" always enters the Stop mode.

When the user programs exceed the permissible execution times only slightly, program scanning can be upheld by modifying these machine data bits, but this may adversely affect the NC functions.

There are two ways of ensuring the executability of large cyclic user programs (max. 4K statements, 8K statements from SW 4) without placing an excessive load on the CPU:

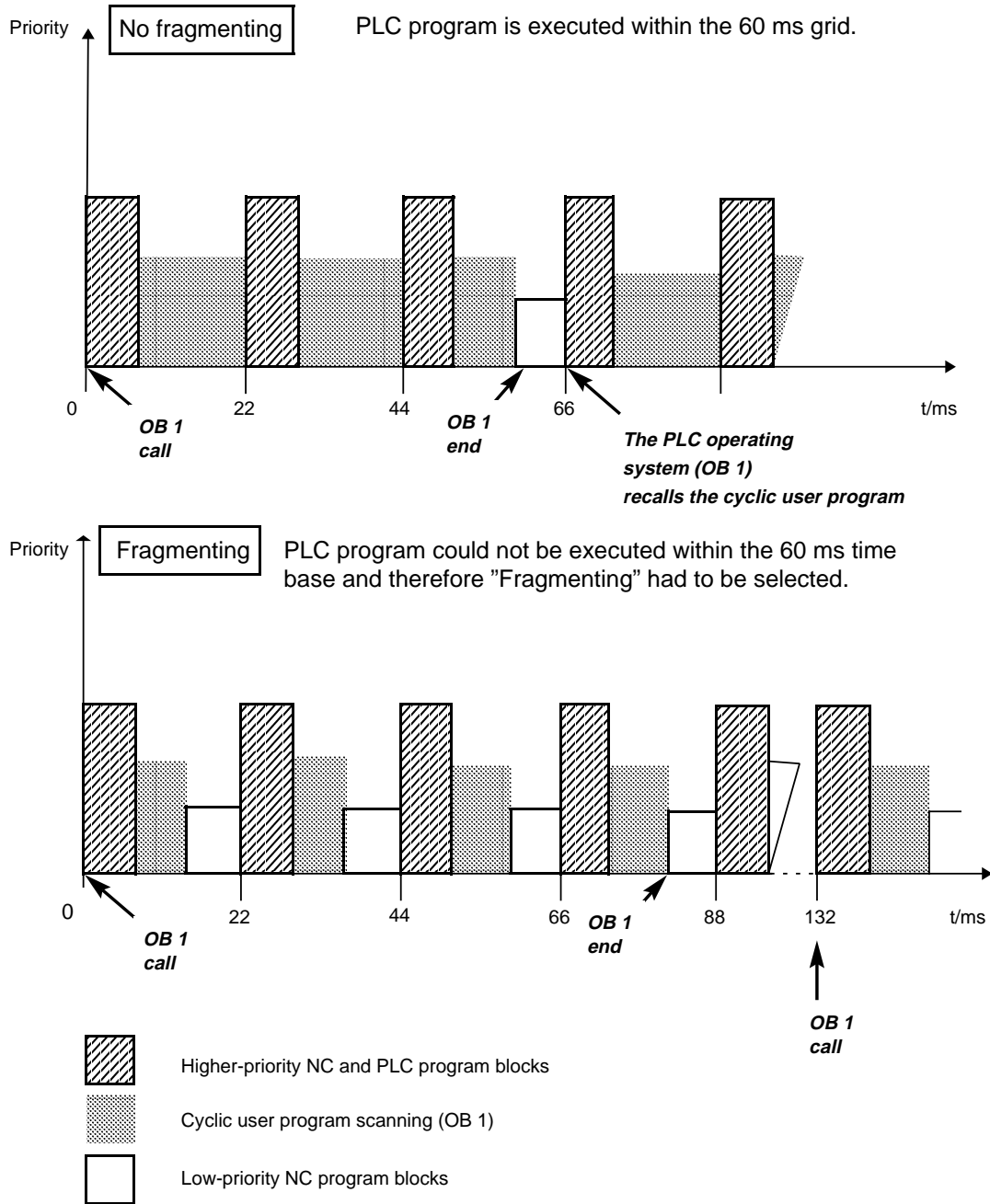
- Fragmenting cyclic execution through autonomous interrupts
- Allowing cyclic execution to be interrupted by higher-priority program blocks only

The mode is set in PLC MD 2003, bit 6. The percentage in PLC MD thus assumes the following meaning as regards the scan time:

- No fragmenting (bit 6 of MD 2003 = 0 = default):
15% of the cyclic program's timing grid (60 ms) results in a permissible runtime of 9 ms.
- Fragmenting (bit 6 of MD 2003 = 1)
15% of 1/3 of the cyclic program's timing grid (60 ms) results in a runtime of 3 ms. The cyclic program then interrupts itself autonomously, and resumes execution in the next third.

Graphic overview of the two methods for cyclic program scanning

Note: OB 1 (cyclical user program) can only be called up in the 60 ms grid.



If the OB 1 (cyclical user program) is too large to be processed in the 60 ms grid, fragmentation must be selected.

The timing grid, of cyclic program scanning is 12 x the scanning time of the position control (12 x 5 ms = 60 ms).

The admissible execution time of the cyclic program for both variants can also be increased by augmenting PLC MD 1 (CPU load in %).

The admissible runtime for the interrupt service routine (PLC MD 3), regardless of the cyclic execution mode, may not exceed 2000 µs in an interval of 4 * scanning time (4 x 5 ms = 20 ms).

The entire runtime may either be used for one-shot execution of the statements in OB 2 or, in an extreme situation, for 4-shot execution of OB 2, as the interrupt input byte is scanned for an edge change on the basis of the scanning time. If the default runtime for OB 2 proves insufficient, PLC-MD 3 must either be incremented (to max. 2500 µs) or the PLC set to the Stop mode when OB 2 exceeds the allotted time by modifying bit 1 in PLC MD 2003. The example on the next page emphasizes the difference between the two variants for a cyclic user program with a runtime time of 25 ms.

No fragmentation:

The cyclic user program may be interrupted by higher-priority NC and PLC program blocks only (e. g. position control, interpolation, monitoring functions, programming and test functions, interrupt service routine (OB 2)). It is otherwise continued until it has terminated and all signals it generated have been forwarded to their destinations (NC, I/Os). Only then can the low-priority functions execute.

Depending on the size of the STEP 5 program in OB 1, the remaining interval may be quite short, as the operating system recalls the cyclic program when 60 ms have passed. The consequence is that too little time is available for the low-priority functions, a fact which becomes immediately apparent in a sluggish "operator interface". Display construction is extremely slow, as is the controller's reaction to keyboard entries. In extreme situations, the low-priority functions are allotted almost no time at all to execute.

In the example, the CPU load caused by the STEP 5 program alone would escalate to value x, where

$$x = \frac{25 \text{ ms}}{60 \text{ ms}} \cdot 100 \% = 42 \% , \text{ assuming that cyclic execution}$$

is terminated within 60 ms. In the example, no regard was paid to the 5% basic load resulting from data transfers prior to and following the OB 1 call.

It is obvious that a CPU load of almost 47% for the controller alone is not acceptable.

Fragmenting:

When this variant is used, the cyclic program interrupts itself autonomously. If the default is used, the CPU load, referred to the cyclic program timing grid, would be:

$$x = \frac{3 \text{ ms} \cdot 3}{60 \text{ ms}} \cdot 100 \% = 15 \%$$

The basic load for the cyclic program, about 5%, must be added to this, so that the cyclic program load on the CPU would always be 20%. It is less than 20% when cyclic execution is not terminated within one timing period. Extremely long cyclic user programs may cause the cycle time monitor to response (default: 300 ms).

Note:

By selecting the diagnostics function (PLC MD 2003, bit 7 = 1), the user can obtain a general view of the anticipated run time for STEP 5 programs as well as the required cycle time. On the basis of these data, he can optimize the configuration of his MD as regards his job requests and the purpose for which the controller is to be used (cf. section 8 "Error Analysis").

1.4 Block overview

Organization data blocks

OB No.	OB designation	OB name
1 2 20		OB for cyclic scanning OB for interrupt scanning OB for one-shot scan after starting
		} User assignable

Program data blocks

PB No.	PB designation	PB name
0 : : 255		User assignable

Function data blocks

FB No.	FB designation	PB name
0 : : 10 11 12 : : 59 60 61 62 63 64 65 66 70 : : 104 105 106 : 199 200 : 255	EINR-DB BLOCK-TR NCD-LESE NCD-SCHR M-STACK STACK-M SEND RECEIVE CONTROL	Reserved : : Reserved Generate data blocks Reserved : : Reserved Blocktransfer Read NC data Write NC data Reserved Reserved Transfer flags to stack Flag stack to transfer flag area Reserved : : } SINEC L2 : Reserved User assignable : User assignable

Data blocks

DB No.	DB designation	DB name
0		Address list block
1		Diagnostics DB
2		User assignable
⋮		⋮
20		User assignable
21		Reserved
⋮		⋮
35		Reserved
36		Interface for data transfer
37		Interface for serial interface control
38		Reserved
⋮		⋮
70		Reserved
71		User assignable
⋮		⋮
⋮		⋮
255		User assignable

Sequence blocks

SB No.	SB designation	SB name
0		User assignable
⋮		
⋮		
255		

2 Organization, Program and Sequence Blocks

2.1 Programming program blocks

The information presented in this Section applies to the programming of organization (**OBs**), program (**PBs**) and sequence blocks (**SBs**). These three block types are all programmed in the same way. Section 3 gives information on programming data blocks and Section 4 information on programming function blocks. Program, organization and sequence blocks can be programmed in all three STEP 5 modes of representation using the basic operations.

The following description deals with program blocks by way of example.

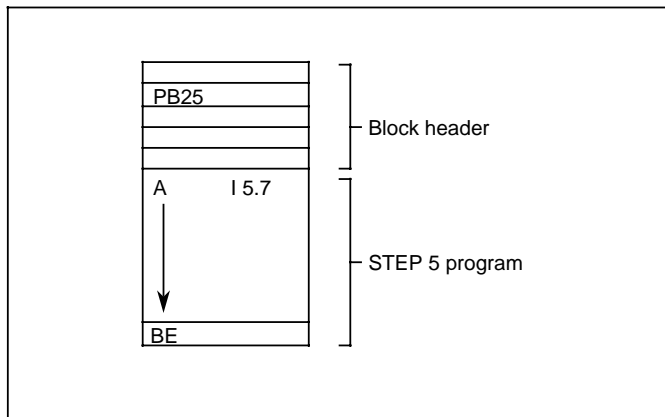
The first step in programming a program block (PB) is the specification of a program block number between 0 and 255 (example: PB 25). This is followed by the actual control program, which is terminated with a "BE" statement.

An S5 block comprises two parts:

- Block header
- S5 operations (block body)

The block header, which the programmer generates automatically, takes up five words in program memory.

A program block should always be a self-contained program. Logical links to other blocks serve no practical purpose.



Structure of a program block

2.2 Calling blocks

Block calls are used to release the blocks for execution. These block calls can be programmed only in organization, sequence, program or function blocks.

(Only organization blocks may not be invoked by the user program).

A block call is comparable to a "subroutine branch", and may be both conditional and unconditional.

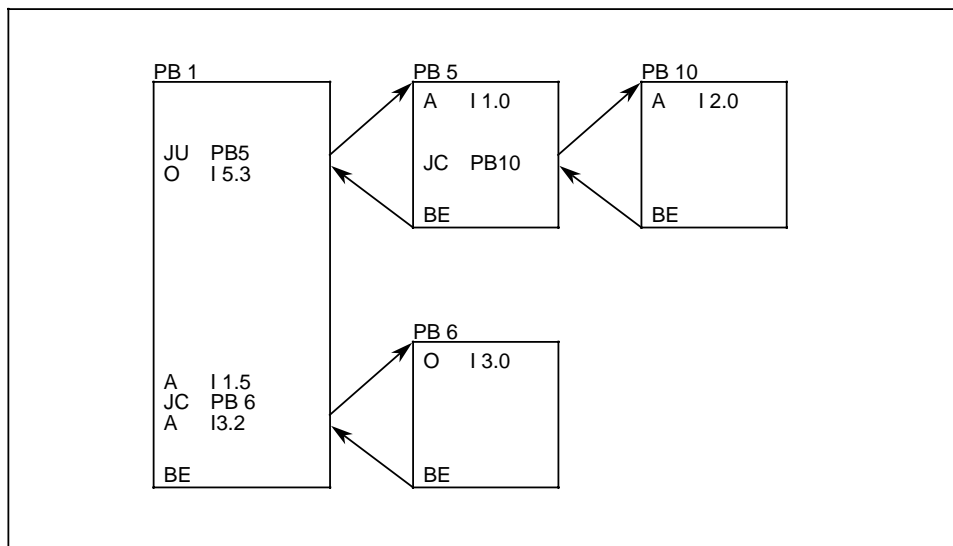
A "BE" statement is used to return to the block that contained the block call. No further logic operations can be carried out on the RLO following a block call or a "BE". The RLO (result of the logic operation) is passed to the "new block", and can be evaluated there.

Unconditional call: for example JU PB5

The program block is executed without regard to the RLO.

Conditional call: for example JC PB6

The program block is executed in dependence on the RLO.



Block calls for enabling execution of a program block

3.2 Calling data blocks

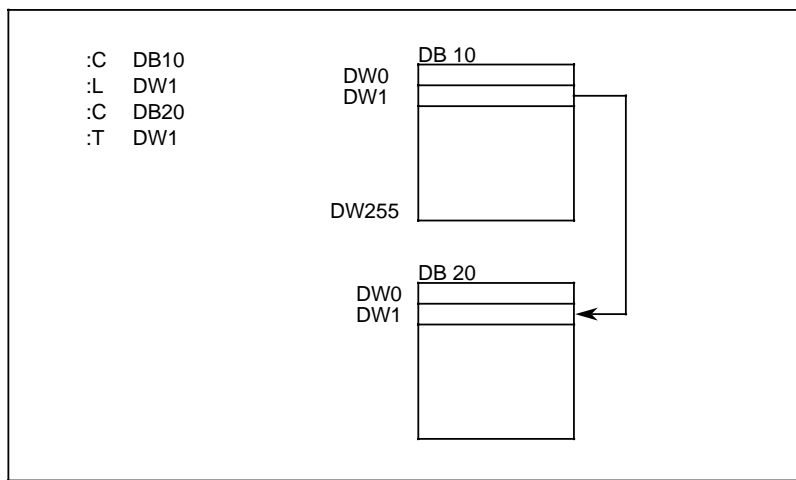
Data blocks can be called unconditionally only. Once called, a data block remains in force until the next is invoked.

User data blocks must not conflict with those required by the system.

A data block call can be programmed in an organization, program, function or sequence block. The "A DB xxx" command calls a data block, e.g. (DB10, i.e. call from DB 10).

Example 1

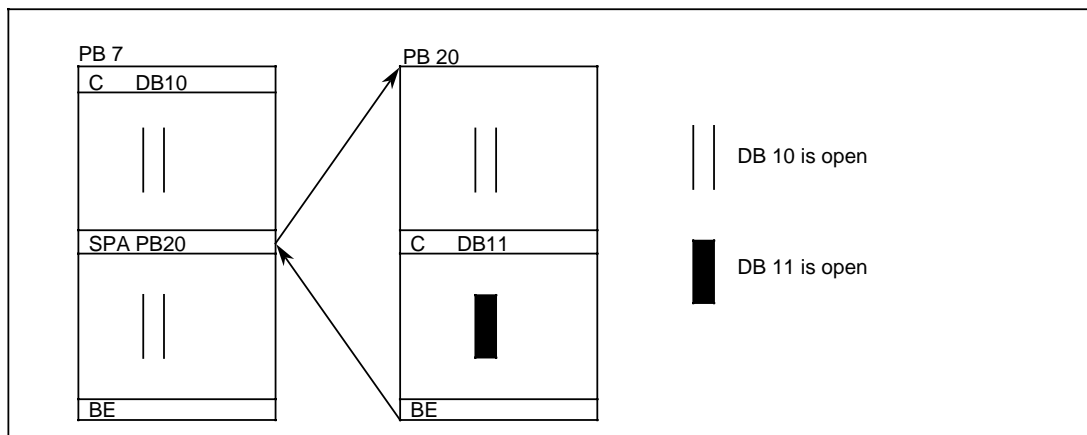
Transferring the contents of data word 1, data block 10 to data word 1, data block 20.



Calling a data block

Example 2

When a program block in which a data block has already been addressed calls another program block that addresses another data block, the latter is valid only in the program block that was called. The original data block is again valid following return to the calling block.



Validity range of a selected data block

3.3 Data blocks created by the operating system

DB 1 : Diagnostics DB

(see Section 8.1)

DB 36 : Status data transfer

DB 36 status data transfer FB 61 / FB 62									
Number Interface Byte	Byte No.	15	14	13	12	11	10	9	8
		Bit No.							
		7	6	5	4	3	2	1	0
1	DL 0	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
2	DR 0	Value 1- Value 3	Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
3	DL 1	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
4	DR 1	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
5	DL 2	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
62	DR 30	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
63	DL 31	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
64	DR 31	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested
65	DL 32	Value 1- Value 3	Error Number format	Message access disabled	Data transfer terminated	Data transfer reserved	Data transfer running	Fifo reserved	Data transfer requested

Note:

- If the PLC goes into stop mode because of a parameter error the number of the interface bytes is entered in the high byte of AKKU 2.
- If several jobs are entered in the buffer for the data lines a job with number 65 is processed before the other jobs.
- Data transfer is performed by function blocks FB 61 (READ) or FB 62 (WRITE).
- In the parameter list a status byte in DB 36 is assigned to every FB 61 or FB 62.

Description of the job-specific interface signals (DB 36)**DATA TRANSFER REQUESTED**

- 1 signal: LESE or SCHR signals of the FBs are on "1".
 0 signal: a) At 0 1 edge of "Data Transfer Terminated" or "Error on data transfer".
 b) LESE or SCHR signals are on "0".

FIFO RESERVED

- 1 signal: Job cannot be entered in FIFO at present. If LESE or SCHR=1 the attempt is repeated until entry is possible.
 0 signal: No effect

DATA TRANSFER RUNNING

- 1 signal: Order entered in buffer or job is being processed.
 0 signal: After 1 signal when DATA TRANSFER TERMINATED has 0 signal.

DATA TRANSFER RESERVED

- 1 signal: FIFO RESERVED, DATA TRANSFER RUNNING signals are on "1".
 0 signal: No effect

Application note: Activation of INPUT DISABLE during data transfer.

DATA TRANSFER TERMINATED

- 1 signal: Job executed without or with NC error message.
 0 signal: After 1 signal when LESE/SCHR has 0 signal on FB 61/62.

MESSAGE ACCESS DISABLED

Available soon

ERROR NUMBER FORMAT

- 1 signal: Job executed with NC error message. FB parameter "Number format" not allowed. (E.g. an axis actual value is to be read as bit pattern.)
 0 signal: After 1 signal only when LESE/SCHR signal is again on "0".

ERROR VALUE 1 - VALUE 3

- 1 signal: Job executed with NC error message. FB parameters WERT 1 to WERT 3 cannot be interpreted by the NC. (E.g. a non-existent machine data is addressed.)
 0 signal: After 1 signal only when LESE/SCHR signal is again on "0".

DB 37: Serial interface

Interface signals								
Byte No.	15	14	13	12	11	10	9	8
	Bit No.							
	7	6	5	4	3	2	1	0
DL 0							V.24 running 2	1
DR 0								

Signals for data transfer initiative PLC								
Byte No.	15	14	13	12	11	10	9	8
	Bit No.							
	7	6	5	4	3	2	1	0
DL 1						V.24 abort	Data start output	Data start output
DR 1							Error on data transfer	Data transfer terminated
DW 2	Datentype for data output							
DW 3	Datentype for data input							
DW 4	Start number							
DW 5	End number							
DL 6								

Data type	DATA TYPE FOR DATA OUTPUT (DB37; DW2, 3)		START NUMBER (DB37;DW4)	END NUMBER (DB37;DW5)	(DB37; DW6)
	(KC)	ASCII code	KF/BCD	KF/BCD	KF/BCD
Part program	MPF	4D50 4620	0 - 9999	0 - 9999	-
Subroutine	SPF	5350 4620	1 - 999	1 - 999	-
Tool offsets	TOA	544F 4120	1 - 99	1 - 99	1
R parameters	RPA	5250 4120	0 - 999	0 - 999	1
NC machine data	TEA1	5445 4131	-	-	-
PLC machine data	TEA2	5445 4132	-	-	-
Zero offsets (G54 - G57)	ZOA	5A4F 4120	-	-	1
NC setting data	SEA	5345 4120	-	-	-

Note:
The ASCII characters DATA TYPE FOR DATA OUTPUT must be entered **left-justified**. The data format for DW 4, 5, 6 is selected by PLC MD 2001.3.

Description of the individual bits (DB 37: serial interface)

On SINUMERIK 805 an interface is assigned by PLC MD 8 (interface for DB 37) which is responsible for serial data transfer (activated by the PLC).

V.24 RUNNING: 1 or 2

D 1.0 and 0.2

1 signal: Data input/output run via interface running.

0 signal: otherwise

Note:

The signal V.24 RUNNING is "1", if data input/output is running via one of the two V.24 interfaces regardless of whether it was activated by the PLC or NC (operator).

DATA START INPUT

D 1.8

1 signal: Start of data input via V.24.

0 signal: By user on:

a) TRANSFER TERMINATED

b) ERROR ON DATA INPUT/OUTPUT

Note:

After start of data input data must arrive at the interface at least within 20 seconds. Otherwise monitoring response (V.24 error).

DATA START OUTPUT

D 1.9

1 signal: Start of data output via V.24.

0 signal: By user on:

a) TRANSFER TERMINATED

b) ERROR ON DATA INPUT/OUTPUT

Note:

Before activation of data output the parameters DATA TYPE, START NUMBER and END NUMBER must be supplied.

V.24 ABORT

D 1.10

1 signal: Abortion of data input/output

0 signal: By user:

a) otherwise

b) TERMINATED ON TRANSFER

DATA TRANSFER TERMINATED**D 1.0**

1 signal: after detection of the end criterion of data input/output

0 signal: a) otherwise
 b) after 1 signal if the signals
 - DATA START INPUT
 - DATA START OUTPUT
 - V.24 ABORT
 carries 0 signal

ERROR ON DATA TRANSFER**D 1.1**

1 signal: after erroneous V.24 activation

0 signal: after 1 signal if the signals
 - DATA START INPUT
 - DATA START OUTPUT
 - ABORT
 carry 0 signal

Note:

The error message can be triggered by:

- a) no or incorrect data at data input
- b) data type for data output incorrect

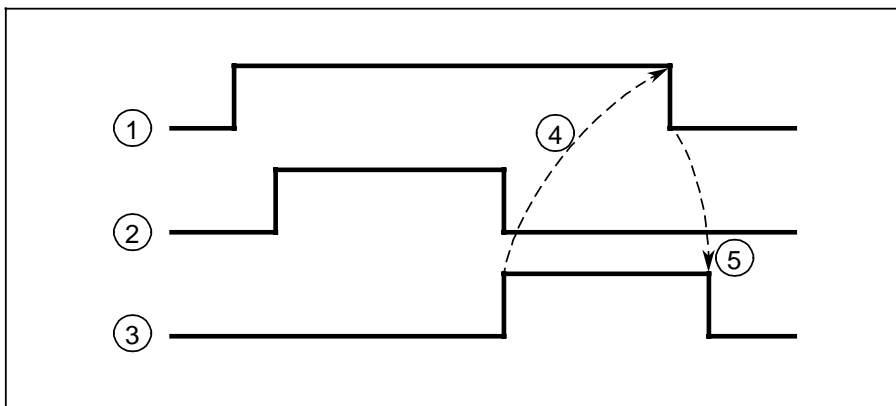
Signal timing:

Fig. 8.1 SP 3568.0

Data input/output without abortion

1. DATA START OUTPUT or INPUT
2. V.24 RUNNING
3. TRANSFER TERMINATED or ERROR ON DATA TRANSFER
4. User deletes signal 1
5. Basic program deletes signal 3

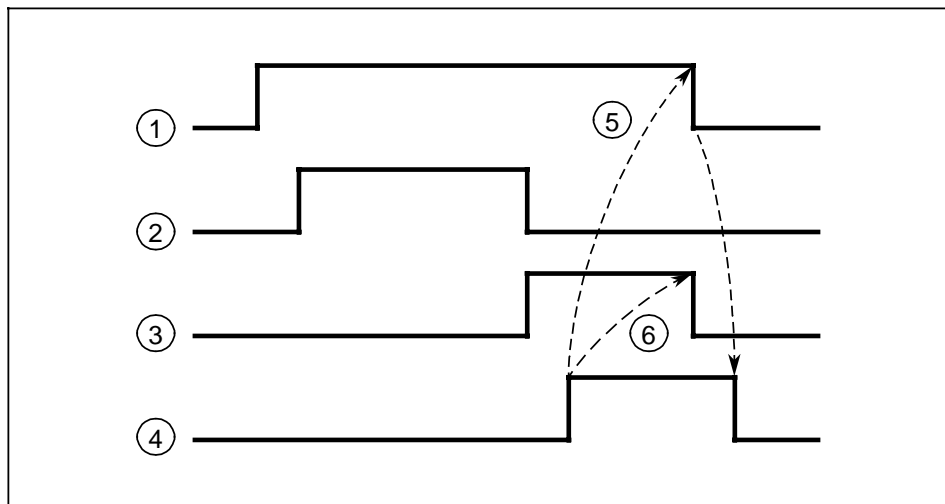


Fig. 8.2 SP 3569.0

Data input/output with abortion

1. DATA START OUTPUT or INPUT
2. V.24 RUNNING
3. V.24 ABORT
4. DATA TRANSMISSION TERMINATED
5. User deletes signal 1 and 3
6. Basic program deletes signal 4

Program example:

Output of part programs 10 to 20. To control transmission (DATA start output and V.24 abort) buttons are used.

```

.
.
.
Q DB 37
L KSMP
T DW 2
L KSF
T DW 3
L KF 10
T DW 4
L KF 20
T DW 5
L KF 1
T DL 6
A I 0.6
S D 1.9
A I 0.7
S D 1.10
A D 1.0
R D 1.9
R D 1.10
    
```

} Load data type "MPF" into DW 2 and DW 3
 } Load start number 10 into DW 4
 } Load end number 20 into DW 5
 } Load 1 into DL 6
 } DATA start output with I 0.6
 } V.24 abort with I 0.7
 } Delete signal DATA start and V.24 ABORT with signal
 DATA TRANSFER TERMINATED

4 Function Blocks

4.1 General remarks

Function blocks are used to implement frequently recurring or extremely complex functions.

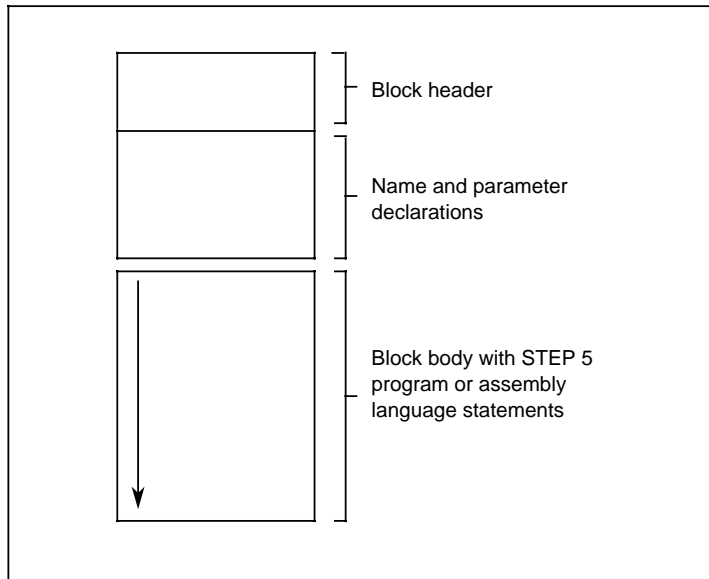
Functions blocks (FBs) are as much a part of the user program as, for example, program blocks. There are three basic differences between function blocks and organization, program or sequence blocks:

- Function block can be initialized, i. e. a function block's formal parameters can be replaced by the actual operands with which the function block is called.
- In contrast to organization, program and sequence blocks, an extended operation set comprising the STEP 5 supplementary operations (see also Section 9.4) can be used to program function blocks, and only function blocks.
- The program in a function block can be generated and logged in statement list form only.

The function blocks in a user program represent complex, self-contained functions. A function block programmed in the STEP 5 language can be programmed by the user himself, or may be purchased from Siemens as a software product. In addition, a number of pretested, technology-specific function blocks can be assembled to form function macros and linked into the basic program. The user can call these macros as he would a function block, but he cannot modify them. These blocks are written in assembly language and are also referred to as "resident" or integral function blocks" (cf. Section 6.1).

4.2 Structure of function blocks

A function block comprises a block header, name and parameter declaration, and the block body.



Structure of a function block

4.2.1 Block header

The block header contains all information which the programmer needs in order to display the function block in graphic form and check the operands when the function block is initialized. The user must enter the header (using the programmer) before programming the function block.

4.2.2 Block body

The block body contains the actual program, i. e. describes the function to be executed in the STEP 5 language. Only the block body is processed when the function block is called.

The programmer echoes the block name and parameter declaration when integral assembly-language function blocks are called.

When the "first executable statement" in the block body is the "ASM" STEP 5 command (switch to assembly code), the processor executes the subsequent assembly language statements immediately.

4.3 Calling and initializing function blocks

Function blocks (FBs) are present only once in memory. They can be called once or more than once by a block, and different parameters can be used for each call.

Function blocks are programmed or called by specifying a block number (FB 0 to 255).

A function block call can be programmed in an organization, sequence or program block or in another function block. A call comprises the call statement and the parameter list.

4.3.1 Call statement

Unconditional call: e.g. JU FB 30

The function block is executed without regard to the RLO.

Conditional call: e.g. JC FB 35

The function block is executed only when the RLO is in signal state "1".

4.3.2 Parameter list

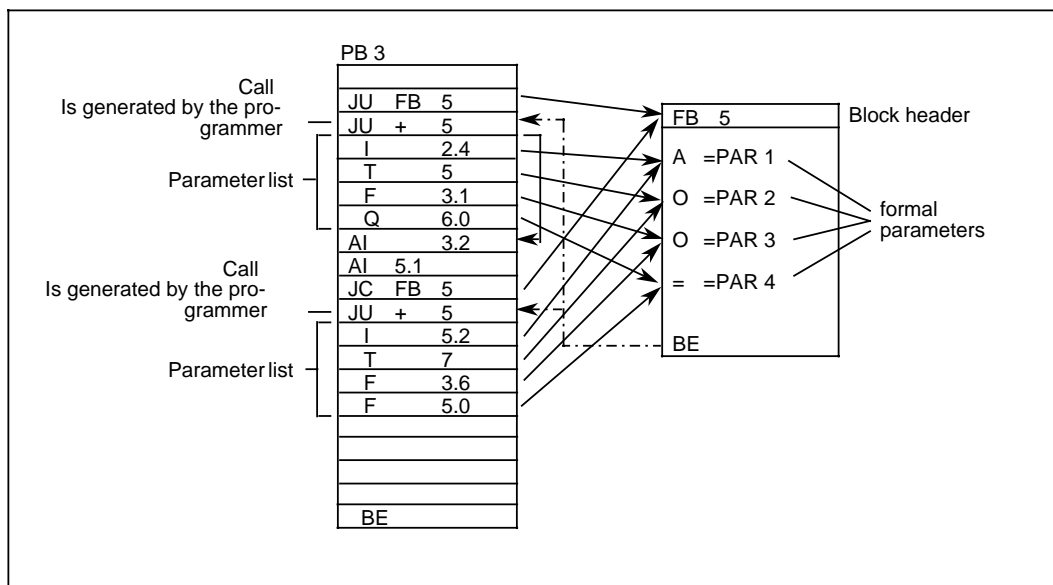
The parameter list immediately follows the call statement, and defines all input variables, output variables and data. The parameter list may contain no more than 40 variables.

The variables from the parameter list replace the formal parameters when the function block is executed. The programmer (PG) monitors the order in which the variables are entered in the parameter list.

The programmer automatically generates, but does not display, the jump statement that follows the FB call.

The FB call reserves two words in program memory, and each parameter one additional word.

The identifiers for the function block's inputs and outputs and the name of the function block are displayed on the programmer when the user programs the function block. This information is in the function block itself. It is therefore necessary that all required function blocks either be resident as function macro in the PLC's basic software, be transferred to the program diskette, or be entered directly into the programmable controller's program memory before function block programming can begin (for details, refer to the Operating Instructions).



Calling a function block

4.4 Programming function blocks

In keeping with its structure, a function block is generated in two parts: the block header and the block body.

The block header must be entered before the block body (STEP 5 program). The block header contains:

- The library number
- The name of the function block
- The formal operands (the names of the block parameters)
- The block parameters

4.4.1 Library number

The library number may be a number between 0 and 65535. The function block is assigned this number without regard to its symbolic or absolute parameters.

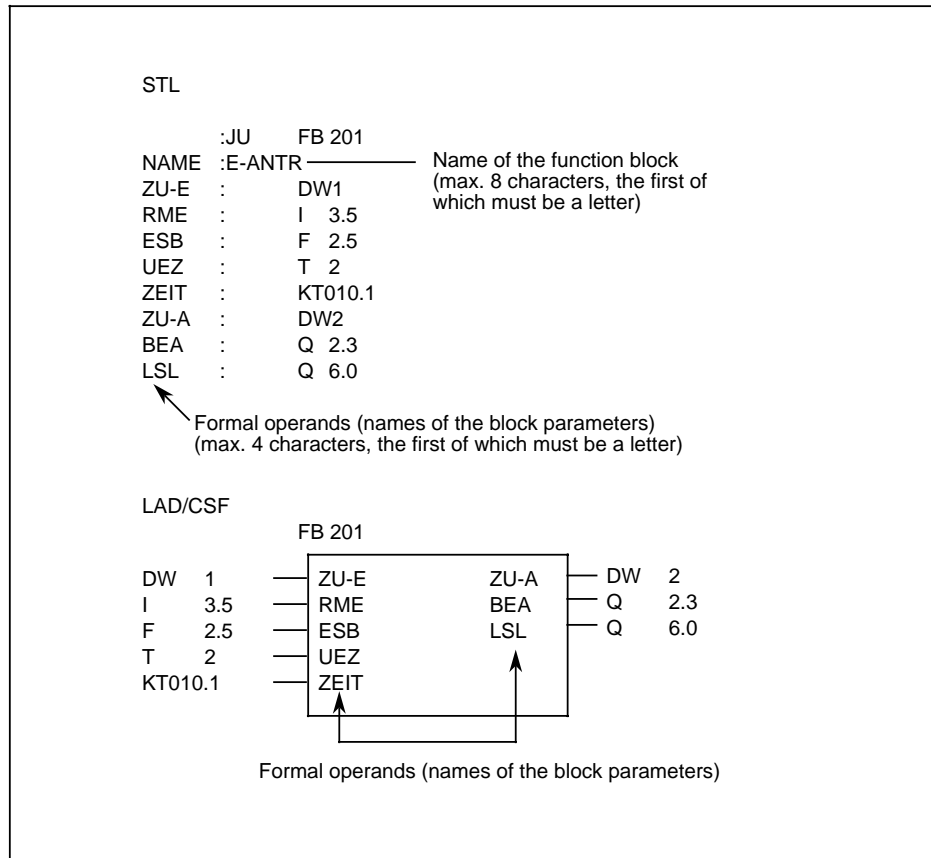
A library number should be assigned only once to permit unique identification of a function block. Standard function blocks have a product number.

4.4.2 Name of the function block

The name that identifies the function block may comprise no more than eight characters, the first of which must be a letter.

4.4.3 Formal operand (block parameter name)

A formal operand may comprise no more than four characters, the first of which must be a letter. A maximum of 40 parameters can be programmed per function block.



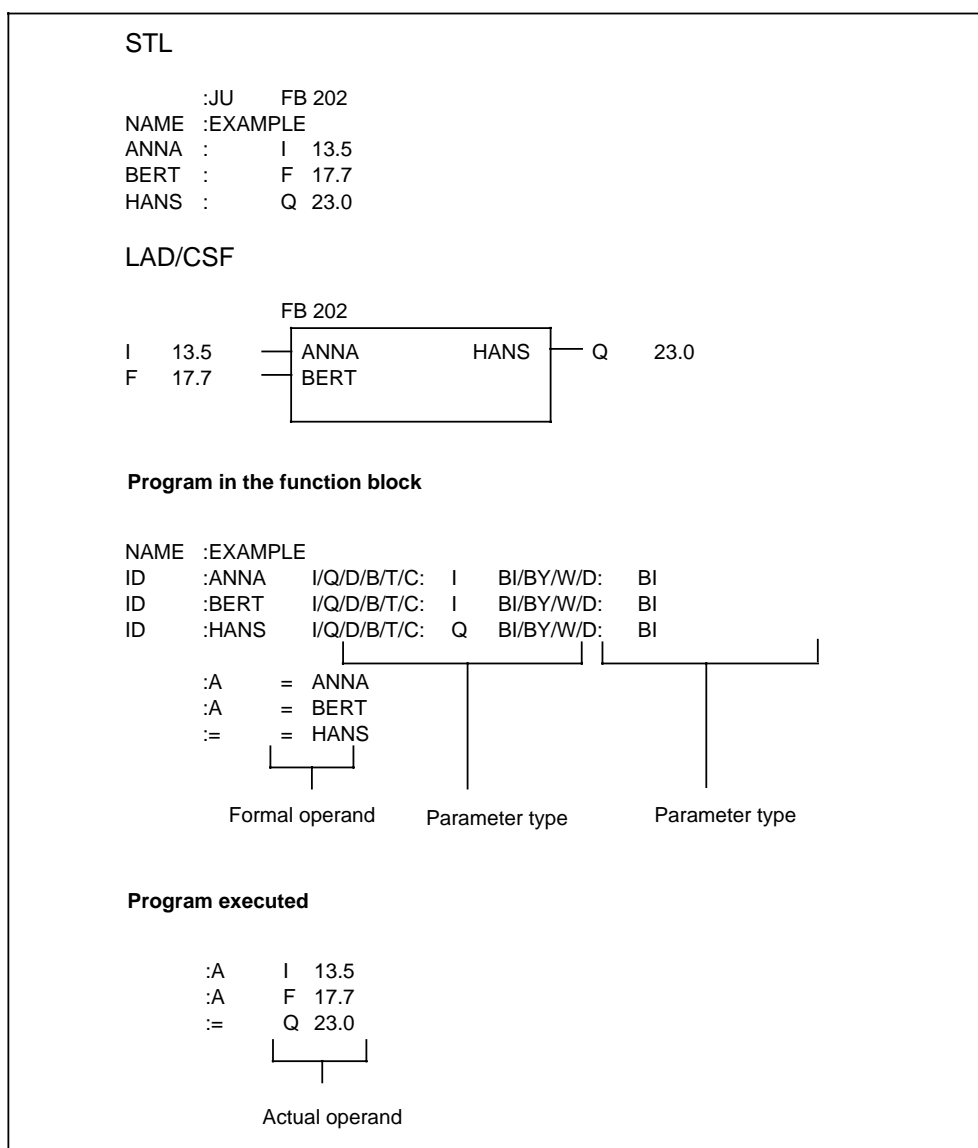
Example: Calling a function block

4.4.4 Block parameter types

A block parameter may be of type "I", "Q", "D", "B", "T", or "C".

- I = Input parameter
- Q = Output parameter
- D = Data
- B = Block
- T = Timer
- C = Counter

In graphic representation, parameters of type "I", "D", "B", "T" and "C" are shown at the left of the function symbol and those of type "Q" at the right. Operations to which parameters are to be assigned (substitution operations) are programmed in the function block with formal operands. The formal operands may be addressed at various locations within the function block.



Example: Calling a function block

4.4.5 Block parameters and permitted actual parameters

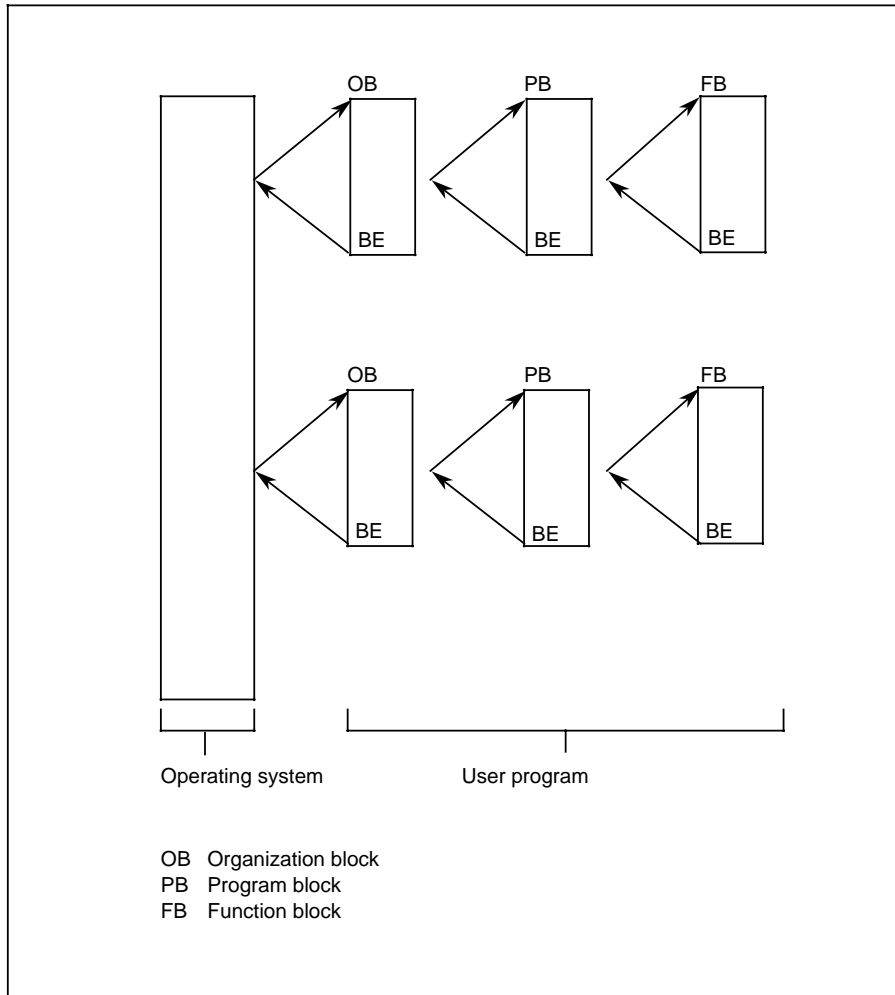
Type of parameter	Type of data	Permitted actual parameters
I, Q	<p>BI for operands with bit addresses</p> <p>BY for operands with byte addresses</p> <p>W for operands with word addresses</p> <p>D for operands with double word addresses</p>	<p>I n.m Input</p> <p>Q n.m Output</p> <p>F n.m Flag</p> <p>IB n Input bytes</p> <p>QB n Output bytes</p> <p>FY n Flag bytes</p> <p>DL n Data bytes left</p> <p>DR n Data bytes right</p> <p>PB n Peripheral bytes</p> <p>IW n Input words</p> <p>QW n Output words</p> <p>FW n Flag words</p> <p>DW n Data words</p> <p>PW n Peripheral words</p> <p>ID n Input double words</p> <p>QD n Output double words</p> <p>FD n Flag double words</p> <p>DD n Data double words</p>
D	<p>KM for a bit pattern (16 bits)</p> <p>KY for two integer numbers expressed in bytes each in the range 0 to 255</p> <p>KH for a hexadecimal pattern (max. 4 positions)</p> <p>KC for a symbol (max. 2 alphanumeric characters)</p> <p>KT for a time (time in BCD) with time base 1.0 to 999.3</p> <p>KZ for a counter value (BCD) 0 to 999</p> <p>KF for a fixed-point number in the range - 32768 to + 32767</p> <p>KG for a floating-point number</p>	Constants
B	Type specification not permitted	<p>DB n Data block, the statement CDBn is executed</p> <p>FB n Function blocks (without parameter)</p> <p>PB n Program blocks are called unconditionally (SPA ..n)</p> <p>SB n Sequence blocks are called unconditionally (SPA ..n)</p>
T	Type specification not permitted	T Timer, the time is specified as data or as a constant in the function block
C	Type specification not permitted	C Counter; the count is specified as data or as a constant in the function block

5 Program Organization

5.1 General remarks

The organization blocks form the interface between the operating system and the user program.

The organization blocks (OBs) are as much a part of the user program as are program blocks, sequence blocks and function blocks, but only the operating system can invoke them. A user can only program organization blocks; he cannot invoke them.



PLC program

Appropriate programming of the organization blocks enables the following:

- One-shot execution following PLC restart (OB 20)
- Cyclic execution (OB 1)
(see page 5-3, "Programming the cyclic program")
- Execution of the interrupt service routine (OB 2)
(see page 5-7, "Programming the interrupt service routine")

OB 2 must be available when interrupt processing is enabled (PLC MD 2002, bit 0=0), otherwise the PLC will stop. OB 20 and OB 1 can be installed in the PLC as required. If one of these two OBs is missing this fact is ignored by the operating system.

The organization blocks are programmed in the same manner as program or sequence blocks, and can be programmed and documented in all three methods of representation (statement list STL, control system flowchart CSF, ladder diagram LAD).

5.2 Cyclic scanning

5.2.1 Interrupt capability

The cyclic user program can be interrupted after each MC5 instruction. The last instruction is executed in its entirety and pending interrupts (caused by a process interrupt or higher-priority sections of the NC program) are enabled when the instruction been executed.

If the interrupt was caused by detection of an edge change in the interrupt input byte, all MC5 registers, as well as flag bytes 224 to 255, are saved; these registers and flag bytes are recovered following termination of the interrupt service routine. The data block open prior to the interrupt is thus once again valid.

5.2.2 Response time and cycle time

- **Response time:**

The response time to a process signal or a signal from the NC is defined as follows: The PLC operating system forwards the signal to the user interface (I, F area), invokes and executes the cyclic user program (OB 1) and transfer the process output image updated by OB 1 to the I/Os and the NC.

The response depends on:

- 1) Type and length of the cyclic user program in OB 1
- 2) Execution mode of the cyclic user program (PLC MD 2003, bit 6: fragmentation, no fragmentation)
- 3) Timing grid for the cyclic user program scanning by the PLC operating system
- 4) Execution time of the higher-priority NC program blocks and of the interrupt service routine (OB 2) where applicable
- 5) Instant: Process image updating by the PLC operating system and actual signal status change (I/Os, internal NC-PLC interface)

- **Cycle time:**

The cycle time is the period between two updatings of the process input image by the PLC operating system. Because the PLC invokes the cyclic user program in accordance with a specific timing grid rather than restarting it immediately after it has terminated, the cycle time can never be less than 60 ms.

Like the response time, the cycle time is dependent on the conditions listed in 1-3 (above). It can be ascertained by invoking the diagnostic function (PLC MD 2003, bit 7 = 1), and is monitored to ensure that it does not exceed a maximum value (PLC MD5 default is 300 ms). If it does, the PLC will stop.

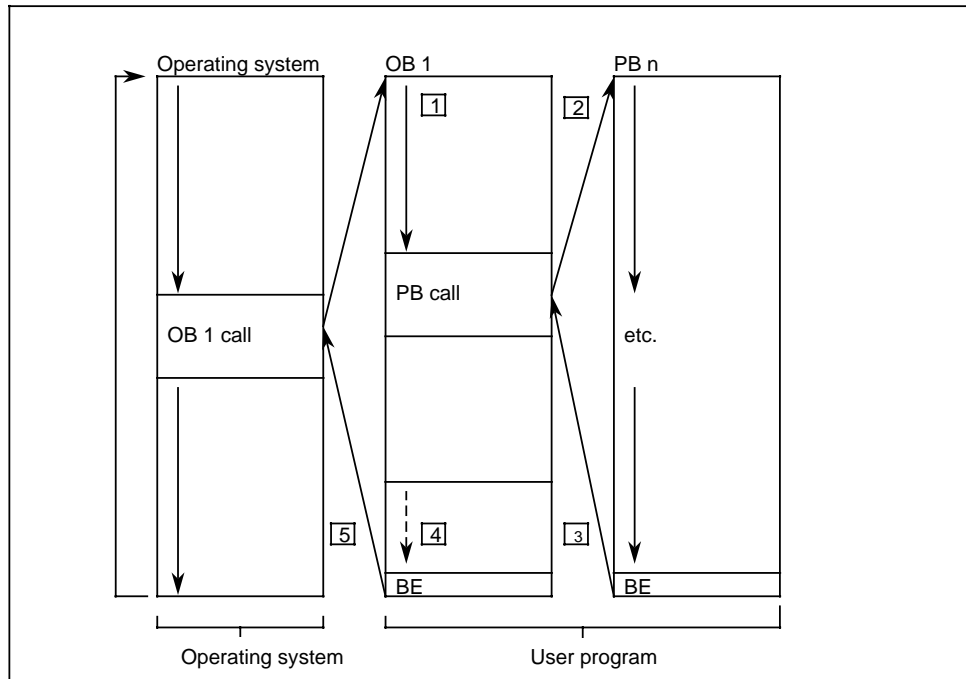
5.2.3 Programming the cyclic program

A programmable controller's program is "normally" scanned cyclically. The processor starts at the beginning of the STEP 5 program, scans the STEP 5 statements sequentially until it reaches the end of the program, and then repeats the entire procedure.

5.2.4 Interface between operating system and cyclic program

Organization block OB 1 is the interface between the operating system and the cyclic user program. The first STEP 5 statement in OB 1 is also the first statement in the user program, i. e. is equivalent to the beginning of the cyclic program.

The program, sequence and function blocks comprising the cyclic program are called in organization block 1. These blocks may themselves contain block calls, i. e. the blocks can be nested (see Section 1, "Program organization").



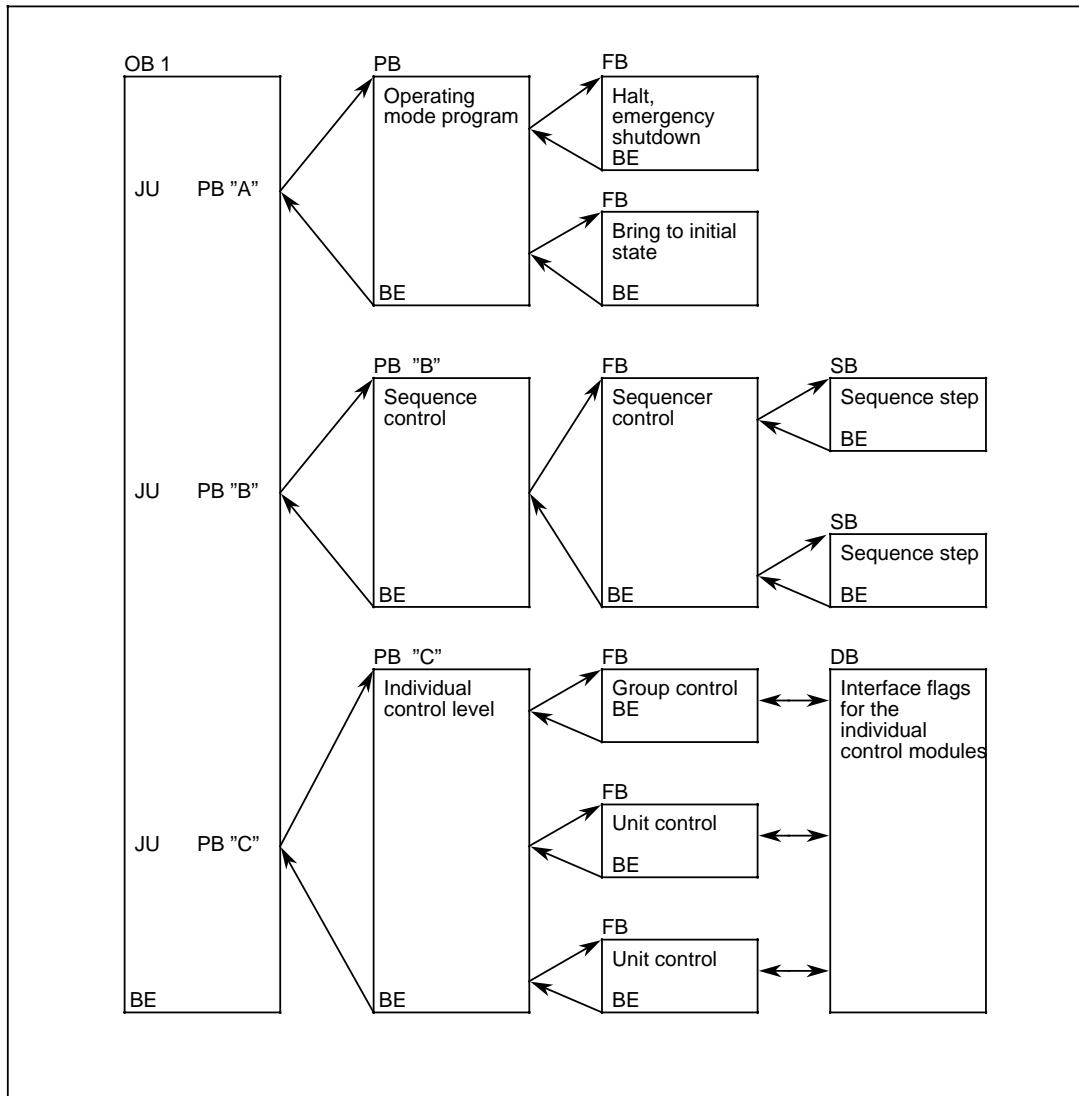
Cyclic program scanning

- 1 First statement in the STEP 5 program.
- 2 First PB call. The block called may contain additional calls (cf. Section 1, "Program organization").
- 3 Return from the last program or function block executed.
- 4 The organization block is terminated with BE.
- 5 Return to operating system.

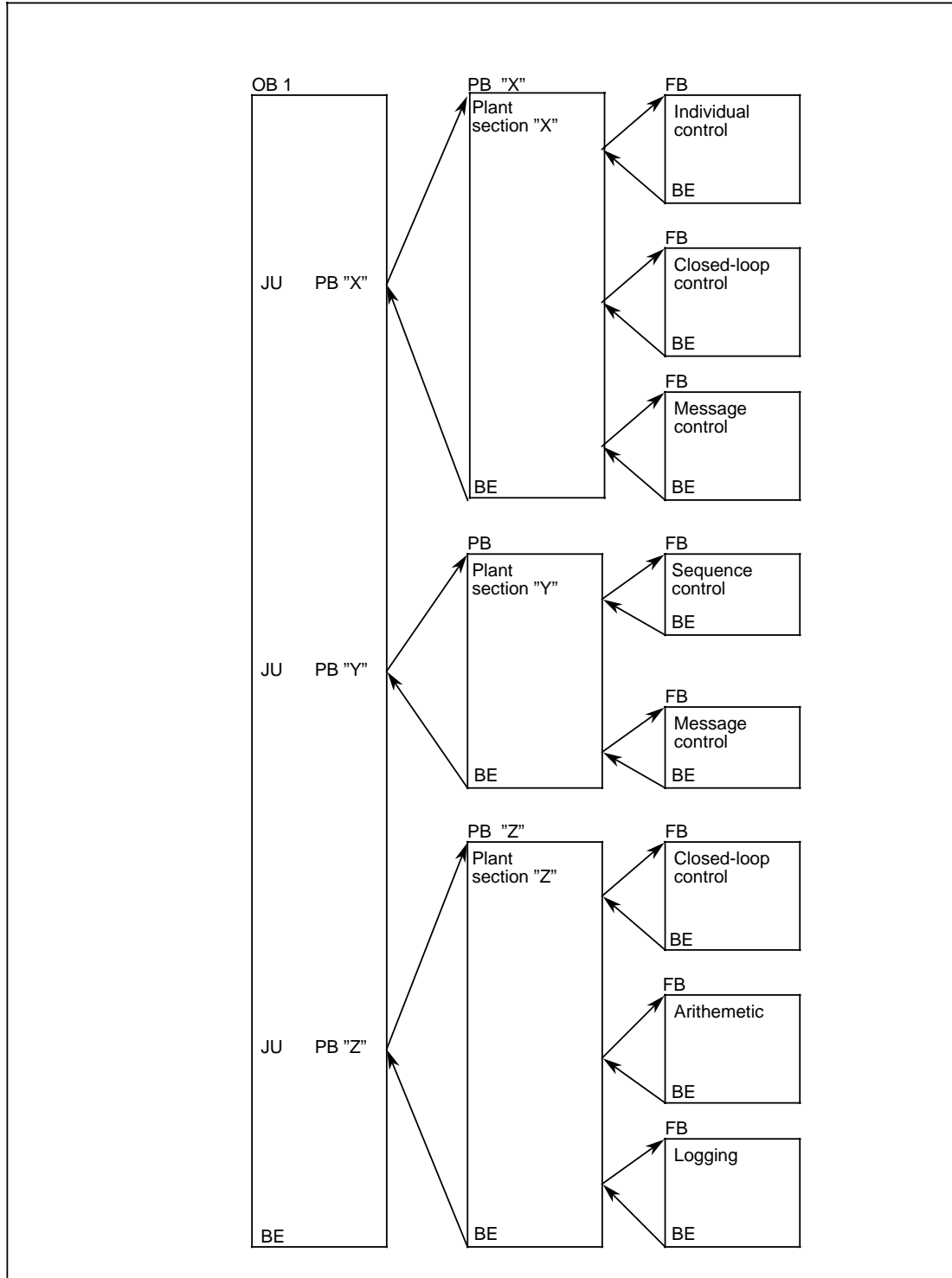
The user program's runtime is the sum of the runtimes of all blocks called. When a block is called "n" times, its runtime must be added to the total "n" times.

5.2.5 Basic program organization

Organization block OB 1 contains the basic structure of the user program. A diagram of this block shows the essential program structures at a glance and emphasizes program-interdependent plant sections.



Breakdown of the user program based on the program structure



Breakdown of the user program based on the plant structure

5.3 Interrupt processing

5.3.1 Programming the interrupt service routine

The PLC has interrupt-processing capabilities.

In this mode, the cyclic program is interrupted and an interrupt service routine (OB 2) executed. Once the interrupt service routine has terminated, the processor returns to the point of interruption and resumes execution of the cyclic program.

The interrupt service routine is initiated when the signal state of a bit of the interrupt input byte (specified in the PLC MD 0) changes.

Interrupt service routines allow the user to react immediately to process signals. An edge change in these signals is thus registered before the process image is updated, thereby minimizing the response time to time-critical functions in the process.

Because the PLC operating system does not transfer the process image to the I/Os when OB 2 terminates, the user must himself influence the process peripherals with STEP 5 commands T PB/T PW.

5.3.2 Interface between operating system and the interrupt

OB 2 is the interface between the operating system and the interrupt service routines.

OB 2 is always invoked when the signal state of a bit in the interrupt input byte changes.

The user may select the input bit via machine data (PLC MD0: maximum value 31). PLC MD 2002, bit 0 must also be set to 0 to enable interrupts.

When one of the selected bits changes from "0" to "1" (positive edge) or from "1" to "0" (negative edge), the interrupt service routine is invoked, i. e. the operating system calls OB 2, which contains the user's interrupt service routine.

The operating system checks the interrupt bytes every 5 ms to determine whether there was a change to a bit in the interrupt input byte during the service routine and invokes an interrupt service routine when required.

The type of signal change (positive or negative edge) is entered as input parameter in flag bytes FB 12 and FB 16.

The PLC operating system resets flag bytes FB 12 and FB 16 when OB 2 terminates.

5.3.3 Response time

The following factors influence the interrupt service routine's response time to an edge change in the interrupt input byte:

- Whether the interrupt input byte is a global or local I/O byte
- The instant at which the edge change takes place in the interrupt input byte and that at which the PLC operating system scans that byte

Approximate response times:

- Global interrupt input byte:
The response time is between 2 ms and 7.5 ms.
- Local interrupt input byte:
The response time is between 3.5 ms and 9 ms.

6 Integral Function Blocks

6.1 General remarks

Function macros are function blocks that are written in assembly language and integrated in the operating system.

The user can invoke and initialize these blocks in the same way as STEP 5 blocks. They are used for time-critical functions, and execute much faster than comparable STEP 5 blocks. The user should therefore give the integral blocks preference over the conventional STEP 5 blocks. When this block is output **only** the block header is displayed.

Note the following when linking the blocks into the user program:

1. These blocks can be called by other blocks when the user program is generated ON-LINE (entered direct in the PLC from the programmer). The corresponding parameter table is displayed on the PG screen, and the user can make all required entries.
2. When programming direct on diskette, the function macros to be called must be disk-resident, i. e. the user must transfer the macros from the PLC to a workdisk. Only the block headers with parameter block are transferred.
3. When these FBs are called in the user program, the user need only initialize the parameter table.


Loading the PLC blocks:

STEP 5 function blocks that have the same numbers as function macros cannot be transferred to the PLC (PG message: "Block already in EPROM", i.e. these function blocks are already contained in the system EPROM).

6.2 FB identifiers

FB call	Parameter type	Data type	Permissible actual operands
	I Input Q Output	BI Operand with bit address BY Operand with byte address W Operand with word address	I n.m Input Q n.m Output F n.m Flag IB n Input byte QB n Output byte FY n Flag byte DL n Data byte left DR n Data byte right PB n Peripheral byte IW n Input word QW n Output word FW n Flag word DW n Data word PW n Peripheral word
	B Block	Not applicable	DB n Data block FB n Function block PB n Program block SB n Sequence block
	T Timer	Not applicable	T n No. of the timer
	C Counter	Not applicable	Z n No. of the counter
	D Data	KM Bit pattern, 16 digits KY Two absolute values (1 per byte) from 0 to 255 KH Hexadecimal number, max. 4 digits KC Two alphanumeric characters KT Time (BCD coded) from 1.0 to 999.3 KZ Count (BCD coded) from 0 to 999 KF Fixed-point number (-32768 to +32767)	
	I, BI I, BI - Q I, BI - /	Static input signal Input signal acknowledged by FB Input signal whose rising edge is evaluated	
	\$I,... \$... *...	DW not allowed as parameter Input signal which must be supplied prior to calling the FB Fixed input signal which need not be generated	
	Q, BI Q- Q, BI I- Q, BI \$...	Static output signal Output signal which must be acknowledged by the user Output signal for one cycle (pulse) Output signal at defined flag word or data word which can be evaluated immediately following the FB	
	*... % 1 %v1	Defined output signal, e. g. NC signal Error code ACCU 2 on system stop (STS); ACCU 1 error code Aux. spec.: No. of the interface byte in ACCU2's HIGH byte	

6.3 Function macros

FB No.	FB ID	FB NAME	Section
11	EINR-DB	Generate data blocks	6.3.1
60	FB BLOCK-TR	Block transfer	6.3.2
61	FB NCD-LESE	Read NC data	6.3.3
62	FB NCD-SCHR	Write NC data	6.3.3
65	M STACK	Transfer flags flag stack	6.3.4
66	STACK M	Flag stack transfer flags	6.3.4
104	SEND	 These function blocks are only available when the "SINEC L2" functionality is activated. See the Function Manual "SINEC L2 Interface Module", order number: 6ZB5 410-0DQ02-0AA0 for details	
105	RECEIVE		
106	CONTROL		

6.3.1 FB 11 EINR-DB Generate data blocks

1. Description

The EINR-DB function block is used to generate data blocks for variable data in the PLC's RAM.

EINR-DB generates the specified data block, e.g. in the cold restart routine (OB 20), but only when the DB is not yet in the address table and a valid data word number is specified. The interface flags (FW 246 and FW 248) are evaluated when parameter AN = 0. The PLC branches into the stop loop if one of the following parameter errors is signaled:

- $255 < \text{DW no.} < 0$
- Data block number > 255
- Data block number = 0
- Not enough RAM available in PLC
- Data block already in PLC and lengths are not identical

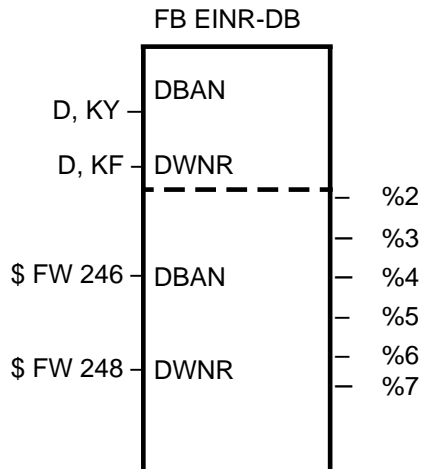
A detailed error identifier is entered in ACCU 2 (ACCU 1 contains the block number). The data block is initialized to zero.

Note: The data blocks for the basic software (interface data blocks) are generated automatically in the cold restart routine.

2. Block data

Bit no.	:
FBs to be loaded	: None
DBs to be loaded	: None
Type of FB call	: unconditional or conditional (JU FB11 or JC DB11)
DBs to be entered	: None
Error messages	: %2 DB no. > 255 %3 Spec. DWNR < 0 %4 Length of DB to be generated not identical with length of DB in PLC %5 Not enough RAM available in PLC %6 Spec. DWNR > 255 %7 DB0 cannot be generated

3. Block call



4. Signals

DBAN Number of data blocks to be generated and their numbers.

Special case:

When parameter AN = 0, the function block can be initialized over flag words FW 246 and FW 248.

High-Byte : DB number

Low-Byte: Number of data blocks (minimum of 1)

DWNR No. of the last DW

5. Example

If DB 150 is not available, it must be generated up to DW 10 on a new start (FB 11 programmed in DB 20).

a) direct parameter assignment

```
OB20
Network 1      0000
0000           :
0001           :
0002           : JU FB 11

0003 NAME     : EINR-DB
0004 DBAN     : KY 150,1
0005 DWNR     : KF +10
0006           :
0007           :
```

b) indirect parameter assignment

```
OB 20
Network 1      0000
0000           :
0001           :
0002           : L      KY
                150,1
0004           : T FW 246
0005           : L KF +10
0007           : T FW 248
0008           : JU FB 11
0009 NAME     : EINR-DB
000A DBAN     : KY 150,0
000B DWNR     : KF +10
000C           :
000D           :
```

6.3.2 FB 60 BLOCK-TR block transfer

1. Description

Function block BLOCK-TR transfers the specified number of data words from a source DB located in RAM to a destination DB in RAM.

The user may specify both the start of the source DB block to be transferred and the start data word in the destination DB.

Prior to initiating the transfer, the PLC checks to make sure that:

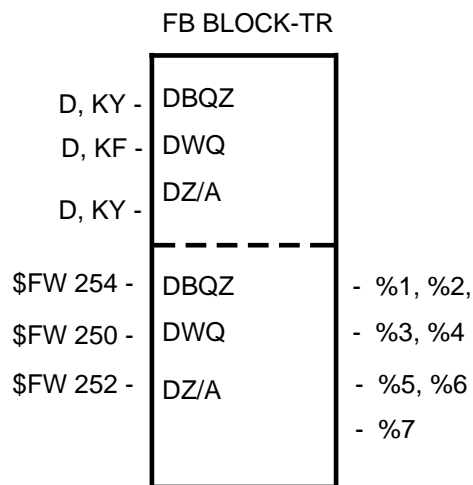
- The source DB in the PLC has the required length
- The destination DB in the PLC has the required length
- The destination DB is located in RAM
- The number of data words to be transferred is > 0 and < 129

The PLC enters the Stop loop when an error is detected. A detailed error identifier is entered in ACCU 2 (ACCU 1 = block number). Flag words FW 250-254 are scanned when the "DB QZ" parameter is zero.
 (Initialization: 0.1 1.0 or 0.0).

2. Block data

- Bit no. :
- FBs to be loaded : None
- DBs to be loaded : None
- Type of FB call : Absolute or conditional (JU FB 60 or JC FB 60)
- DBs to be entered : None
- Error messages : %1 No. of DWs to be transferred > 128
 %2 No. of DWs to be transferred = 0
 %3 No source DB
 %4 No destination DB
 %5 Destination DB too short
 %7 Source DB too short

3. Block call



4. Signals

DBQZ Source DB and destination DB numbers

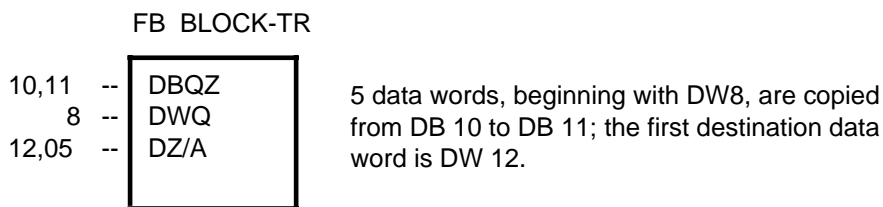
High byte : Source DB
Low byte : Destination DB

DWQ Start of the data block to be copied in the source DB

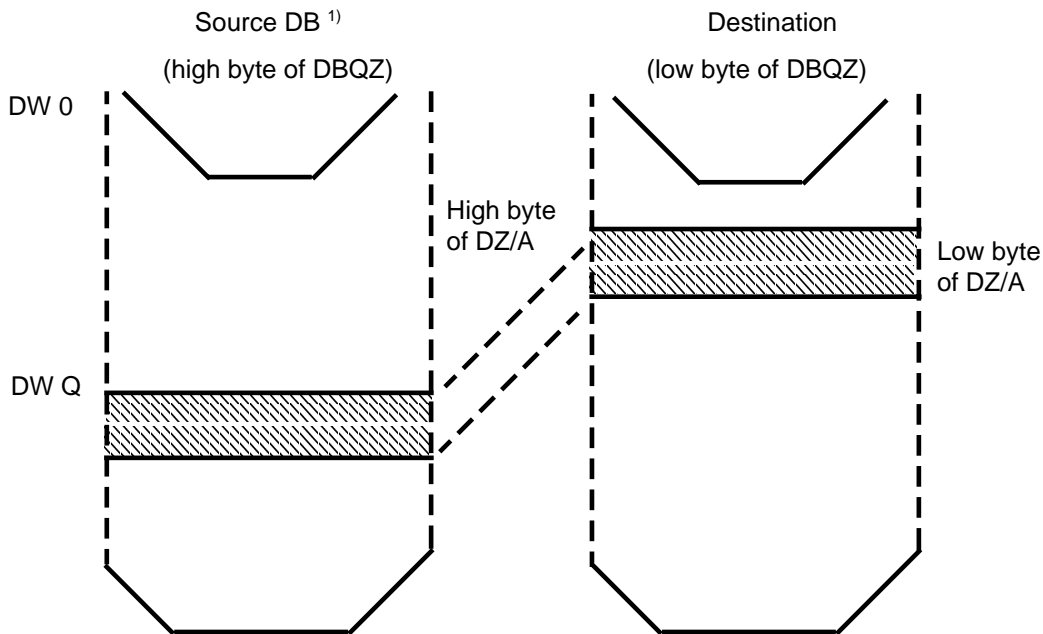
DZ/A Start of the data block in the destination DB and number of data words to be transferred

High byte : Start of the data block
Low byte : Number of data words to be transferred

5. Example



6. Data transfer diagram



1) The "source" data block may be located in either RAM or EPROM

6.3.3 FB 61 NCD-LESE Read NC data FB 62 NCD-SCHR Write NC data

1. Description

Function blocks FB 61 and FB 62 are used to read NC data from/write NC data to the PLC. Prerequisite is that the job-controlled data interchange between NC and PLC has been enabled by NC MD 5015 bit 1 (external data input).

Parameters must be initialized to inform the function blocks of the data source in the NC or PLC and of the data destination in the PLC or NC. The function blocks must be assigned a byte in the interface data area (DB 36) over the NBSY parameter; this byte informs the function blocks of the current state of the data transfer.

The NC/PLC data transfer function is described in detail in Section 7. NC interrupt 3016 is issued when the NC's data type is unknown.

2. Block data

FBs to be loaded : None
DBs to be loaded : None
Type of FB call : Unconditional or conditional
DBs to be entered : None
Error messages : ACCU 1 (FB no.) = 61 or 62
ACCU 2, high byte, = no. of the interface byte, i. e. the number of the job that produced the error (exception: error 8).

ACCU 2 low byte = detailed error code:

- 0 : ANZ >1 not allowed
- 1 : Invalid interface byte
- 2 : Addressed data word missing / DB missing or DB no. / FW no. invalid
- 3 : Invalid data type
- 4 : *ANZ =< 0 or >80
- 5 : Illegal read / illegal write
- 6 : Invalid number format
- 7 : Value 3 (WER3) not 0 (ZOA) or 1 (ZOFA)
- 8 : Read/write NC data not enabled (option)

* Refer to the exceptions for the ANZ parameter

3. Block call

FB 61: NCD-LESE

I, BI	--	LESE	
I, BY	--	NSBY	
D, KF	--	ANZ	
D, KS	--	DTY1	
D, KS	--	DTY2	
D, KS	--	DTY3	
D, KF	--	WER1	
D, KF	--	WER2	
D, KF	--	WER3	
D, KS	--	ZFPN	
D, KY	--	ZIEL	
<hr/>			
\$FB 242	--	NSBY	-- % 0
\$FB 243	--	ANZ	-- % 1
\$FW 244	--	WER1	-- % 2
\$FW 248	--	WER2	-- % 3
\$FW 250	--	WER3	-- % 4
\$FW 252	--	ZFPN	-- % 5
\$FW 254	--	ZIEL	-- % 6
			-- % 7

FB 62: NCD-SCHR

I, BI	--	SCHR	
I, BY	--	NSBY	
D, KF	--	ANZ	
D, KS	--	DTY1	
D, KS	--	DTY2	
D, KS	--	DTY3	
D, KF	--	WER1	
D, KF	--	WER2	
D, KF	--	WER3	
D, KS	--	ZFPN	
D, KY	--	QUEL	
<hr/>			
\$FB 242	--	NSBY	-- % 0
\$FB 243	--	ANZ	-- % 1
\$FW 244	--	WER1	-- % 2
\$FW 248	--	WER2	-- % 3
\$FW 250	--	WER3	-- % 4
\$FW 252	--	ZFPN	-- % 5
\$FW 254	--	QUEL	-- % 6
			-- % 7
			-- % 8

4. Signals

LESE The parameters are transferred to the FIFO buffer (job buffer) when the LESE (read) or **SCHR** (write) signal = 1.
 In case of an unconditional block call, the user has to reset the LESE or SCHR signal at the end of the data transfer.
 On an unconditional block call, the signal must be set to "1" (F 0.1) (see also timing diagrams). Data transfer can be activated only when the initialized interface byte's (NSBY's) DATA TRANSFER ENABLED signal is 0.

NSBY The function block must be assigned a byte in the interface data area (DB 36) from which it can ascertain the current state of the data transfer.

Exception: Variable initialization; refer to the overview on the next page.

Note:

When byte DL32 (= number 65) is used as interface byte, the specified job request is serviced on an "interrupt-driven" basis, i. e. it is "sandwiched" between the job requests that are already in the request buffer. With job number 65 and ANZ > 1, the interrupt job is executed fully.

ANZ Number of data words to be transferred. If more than one word is transferred (ANZ > 1), both source and destination addresses are incremented. If this results, for example, in addressing of a data word that is no longer available, the PLC goes to STOP with % 2 (ACCU 2).
Note that a different number of data words is required for each value transferred, depending on the ZFPN parameter.

Exceptions:

When ANZ is 0 or 128, the NSBY, ANZ, WER1-WER3, ZFPN and ZIEL/QUEL parameters are initialized over flag words. The information presented in 5. below (Variable initialization) discusses the difference between ANZ = 0 and ANZ = 128.

DTY1 2 to 6 ASCII characters; CL800 mnemonics (cf. table, para. 7).

DTY2 The values must be input, beginning with DTY1.

DTY3 Missing characters must be filled out with blanks.

WER1 Values entered according to data type (see following table, section 7)

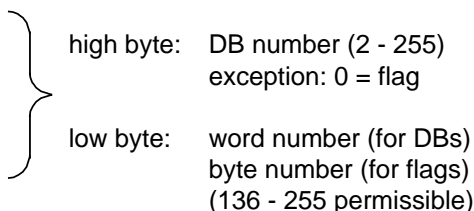
WER2 Example: R parameter R897, data type RPNC, WER1=1, WER2=897

WER3 See following table, section 7

ZFPN PLC/NC number format
Overview of PLC/NC number format is in section 6

ZIEL Data destination in the PLC (FB 61)

QUEL Data source in the PLC (FB 62)



Note:

1. The number of bytes required must accord with the ZFPN parameter.
2. If, for example, a 16-bit word is to be passed to the NC, the value, when transferred, must be written from data words into the initialized word k + 1 when number formats FO-FF are needed.
The parameterized word k has to be set to 0 (see para. 6).

5. Indirect initialization

When the indirect initialization option is used, the values must be written into the flag words in the same format as for direct initialization of the FB.

Example:

The following must be programmed to initialize the ZFPN parameter with F1:

```

.
.
L   KSF1
T   FW252
.
.
.

```

Exception:

The NSBY parameter can be initialized in one of two ways.

- When ANZ = 0, the number of the interface byte must be entered in **FB 242** (refer to Section 1.5, Interface Description).

No. of the interface byte		Byte number
1	≙^	DLO
2	≙^	DRO
.		.
.		.
64	≙^	DR31
65 (interrupt-driven)	≙^	DL32

Example:

The following must be programmed when DL15 is to be used as NSBY:

```

:
L   KB31
T   FB 242

```

- When ANZ = 128, the OP-code¹⁾, with parameters from L DL xx/L DR xy, must be entered in **FW 241**. This method of initialization is particularly suitable, for example, when FB 61 is called as a subroutine of a function block or when NSBY parameter was declared in "I/BY" data format in the calling FB. The programmer generates the right code in the parameter table when the calling FB is initialized, thus enabling it to be passed to FB 61/FB 62 with the statement sequence.

```

:
LW =   ABCD is a parameter (data type: I/BY) of the calling FB.
T   FW 241
:

```

Example OP-code:

No. of the interface byte	Byte No.	OP-code (hexa)
.		
.		
.		
31	DL15	220F
32	DR15	2A0F

6. Overview of NC / PLC number formats

The format of the numbers to be passed to the PLC is specified with the "Number format" parameter. The number formats permissible for all data types are listed in the table shown under para. 7.

The table below presents an overview of permissible parameters.

ZFPN	Description
B0/F0	Value without decimal point (e.g. 1234)
B1/F1	Value with decimal point (e.g. 1234.)
B2/F2	1 decimal place
B3/F3	2 decimal places
B4/F4	3 decimal places
B5/F5	4 decimal places
B6/F6	5 decimal places
B7/F7	6 decimal places
B8/F8	7 decimal places
B9/F9	8 decimal places
FA	Linear-axis position value *
FB	Rotary-axis position value *
FC	Linear-axis feedrate *
FD	Rotary-axis feedrate *
FE	Rotational feedrate *
FF	Rotational speed value *
BG	Value as stored
BI	Bit pattern
AS	ASCII string, 28 bytes long
WB	Weekday in letters (for DTY1-3=DATUHR)

- * Depending on the input system (which is set via machine data), these number formats determine only **the position** of the decimal point within the R parameter value without physical unit.
No checks are made for range violations.

6.1 Memory requirements of the number formats

B0...B9, BG BCD number

	DL		DR	
DWn/MWn	unassigned		Sign: 1	8
DWn+1/MWn+2	7	6	5	4
DWn+2/MWn+4	E	3	2	1

Sign=1 neg. BCD number
 Sign=0 pos. BCD number

Example:

R100 = - 87654.321
 ZFPN parameter = B4

Note:

The first letter in the parameter (in this case B) indicates that the data source or data destination in the PLC is in BCD. The second character in the parameter (4 in this case) indicates that the BCD number is stored in the PLC with three decimal places (FB 61).

On a data transfer from PLC to NC (FB 62), the parameter specifies the location of the decimal point in the NC, regardless of where it is located in the PLC.

If PLC data words contain the value 1234.45, the ZFPN parameter is initialized to B2 for a transfer from PLC to NC, and the number 1234.5 entered in R100.

Exception:

When reading/writing date/time with FB 61/62 you require four data/flag words for number format B0.

Transfer date/time

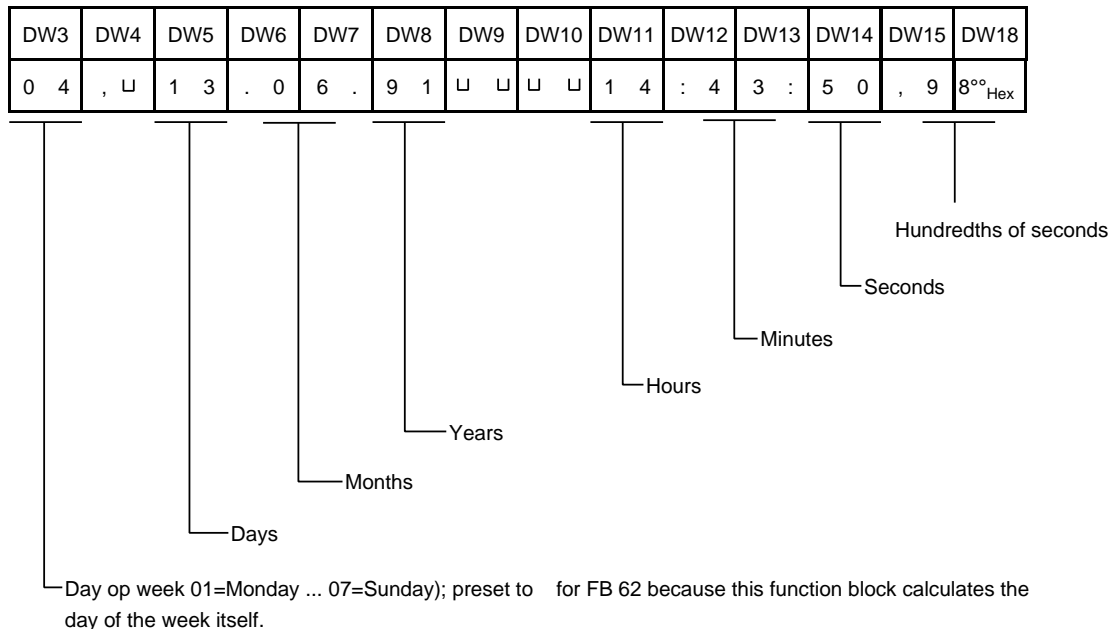
	DL	DR
DWn/MWn	Day of the week	Day
DWn+1/MWn+2	Month	Year
DWn+2/MWn+4	Hours	Minutes
DWn+3/MWn+6	Seconds	1/100 seconds

Example of transfer of date and time, destination/source is DB 200 from DWO:

Date: Friday, 27.04.1990 Time: 14:36:20

	DL	DR
DB 200/DW 0	06	27
DB 200/DW 1	04	90
DB 200/DW 2	14	36
DB 200/DW 3	20	00

The ASCII string from DW3 onwards is shown.



WB Day of the week in letters

Like the AS number format, this number format also supplies a 28 byte long ASCII string. The only difference is that the day of the week is coded using two letters instead of two numbers.

Day of the week: Mo = Monday (German: Montag)
 Di = Tuesday (German: Dienstag)
 Mi = Wednesday (German: Mittwoch)
 Do = Thursday (German: Donnerstag)
 Fr = Friday (German: Freitag)
 Sa = Saturday (German: Samstag)
 So = Sunday (German: Sonntag)

Preset the day of the week to for FB 62 because this function block calculates the day of the week itself.

7. Table for data transfer NC/COM PLC data words/flags

Functional description	Data type (DTY1-DTY3)	Range	Value (WER1 - WER3)	Number format (ZFPN) FB 61 *1	Number format (ZFPN) FB 62 *1	Max. value of parameter ANZ *2
Machine data						
Machine data NC	MDN <Address>	0...4999	1	B0,F0 (*3)	B0,F0 (*3)	80
Machine data NC bytes	MDNBY <Address>	5000...9999	1	BI	BI	80
Setting data						
Setting data NC	SEN <Address>	0...4999	1	B0,F0 (*3)	B0,F0 (*3)	80
Setting data NC bytes	SENBY <Address>	5000...9999	1	BI	BI	30
Tool offsets						
Tool offset	T0S			B0-B9,F0-F9,	B0-B9,F0-F9,	
	<D no.>	1 - 99	1	BG, BI	BG, BI	
	<P no.>	0 - 9	2			10
Tool offset add.	T0A				B0-B9, F0-F9	
	<TO range>	0	1			
	<D no.>	1 - 99	2		BG, BI	
	<P no.>	0 - 9	3			10
Zero offsets						
Settable zero offset (G54 - G57) coarse/fine	Z0A <Group>	1 - 4	1	B0-B9,F0-F9	B0-B9,F0-F9	1
	<Axis no.>	1 - 4	2		BG	
	<c/f>	0/1	3			
Programmable zero offset (G58, G59)	Z0PR <Group>	1 - 2	1	B0-B9,F0-F9	B0-B9,F0-F9	1
	<Axis no.>	1 - 4	2		BG	
Settable zero offset, additive, write only, coarse/fine	Z0FA <Group>	1 - 4	1		B0-B9,F0-F9	1
	<Axis no.>	1 - 4	2		BG	
	<c/f>	0/1	3			
External zero offset from PLC	Z0E <Axis no.>	1 - 4	1	B0-B9,F0-F9	B0-B9,F0-F9	1
PRESET offset	Z0PS <Axis no.>	1 - 4	1	B0-B9,BG,F0-F9	B0-B9, BG,F4	1
Total offset	Z0S <Axis no.>	1 - 4	1	B0-B9,BG,F0		1

*1 The relevant FB cannot execute a data transfer unless a number format has been specified

*2 For data types with more than one parameter, the number of parameters is located in the line containing the WERT value parameter that is incremented.

A number > 1 is possible only under the following conditions:

- Data block in the NC is **closed**
- PLC source or destination address of sufficient length
Three words per value for B0/B9/BG
Two words per value for F0-FF
One word per value for BI

*3 Input/output is carried out in accordance with the format of the machine/setting data specification.

Functional description	Data type (DTY1-DTY3)	Range	Value (WER1 - WER3)	Number format (ZFPN) FB 61 *1	Number format (ZFPN) FB 62 *1	Max. value of parame- ter <u>ANZ</u> *2
Actual values						
Axis position Workpiece-related	ACPW<Axis no.>	1 - 4	1	B0-B9/BG, F0		1
Axis position Machine-related	ACPM<Axis no.>	1 - 4	1	B0-B9/BG, F0		1
External setpoints						
External contour feedrate	EXBF <1> <lin/rot>	1 0/1	1 2	B4, F4	B4, F4	1
Program data						
Program pointer for current block	PP <1> <Level>	1 0 - 3	1 2	B0, F0		1 *4
Program selection						
Selection of an NC program	INITMP <1>	1	1		B0, F0	1
Selection of an NC subroutine	INITSP <1>	1	1		B0, F0	1
R parameters						
R parameter	RPNC<1> <Paramet.>	1 0 - 999	1 2	B0-B9,F0-FF BG, BI	B0-B9,F0-FF BG, BI	80
Time of day						
Data and time of day	DATUHR <1>	0	1	B0, AS WB	B0, AS WB	1

*1 The relevant FB cannot execute a data transfer unless a number format has been specified.

*2 For data types with more than one WERT (value) parameter, the number of parameters is in the line containing the WERT parameter that is incremented.

A number > 1 is possible only under the following conditions:

- Data block in the NC is **closed**
- PLC source of destination address of sufficient length:
Three words per value for B0-B9/BG (for time of day 4 data word per value)
Two words per value for F0-FF
One word per value for BI

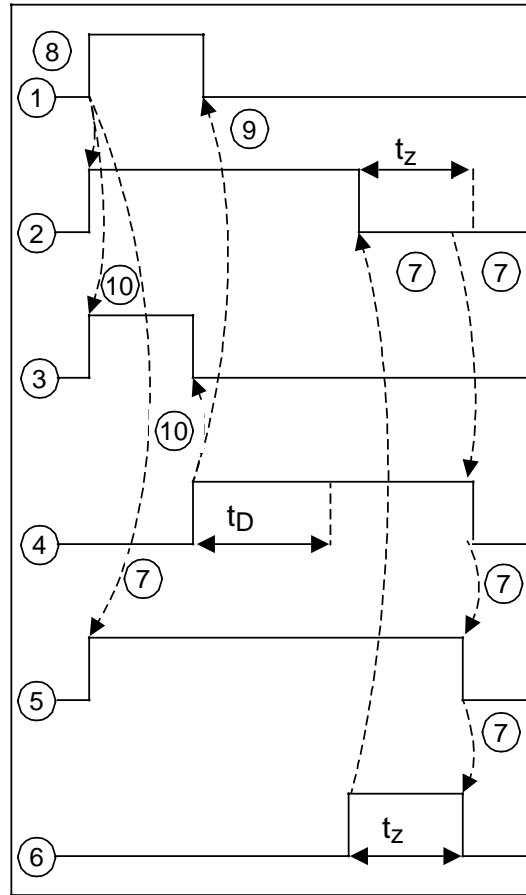
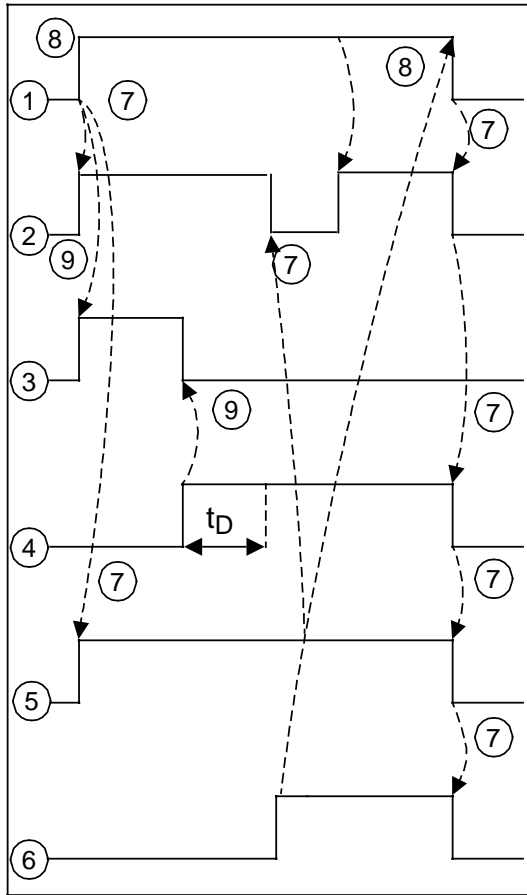
*3 The value is passed as it is defined internally on the basis of the NC machine data for input resolution.

*4 Three items of information are provided on reading out the program pointer when ANZ = 1:

- Level 0: Program type (0 = no program selected, 1 = main program, 2 = subroutine), program number and block number
- Level 1 and up: Subroutine number, number of passes and block number

8a) Timing diagram for the interface signals

8b) Conditional block call



- 1 : READ/WRITE
- 2 : DATA TRANSFER REQUESTED
- 3 : FIFO FULL
- 4 : DATA TRANSFER IN PROGRESS
- 5 : DATA TRANSFER BLOCK BUSY
- 6 : DATA TRANSFER TERMINATED
+ ERROR, if any
- 7 : Signal change initiated by FB
- 8 : Signal change initiated by user
- 9 : Signal change initiated by FB,
not applicable if FIFO not yet full
- t_D : Data transfer block is using
internal interface

- 1 : READ/WRITE
- 2 : DATA TRANSFER REQUESTED
- 3 : FIFO FULL
- 4 : DATA TRANSFER IN PROGRESS
- 5 : DATA TRANSFER BLOCK BUSY
- 6 : DATA TRANSFER TERMINATED
+ ERROR, if any
- 7 : Signal change initiated by FB
- 8 : Signal change initiated by user
- 9 : User no longer needs block
- 10 : Signal change initiated by FB,
not applicable if FIFO not yet full
- t_z : PLC cycle time
- t_D : Data transfer block is using
internal interface

9. Sample parameters for FB61 / FB62

Example 1: The content of the R parameter R 15 is stored in DB 150 from DW 1 by I 5.7 (R15= 258.365).

0066	:		DB150
0067	:	A I 5.7	
0068	:	JC FB 61	0: KH = 0000;
0069 NAME	:	NCD-LESE	1: KH = 0000;
006A LESE	:	F 0.1	2: KH = 0258;
006B NSBY	:	DL 5	3: KH = E365;
006C ANZ	:	KF +1	4: KH = 0000;
006D DTY1	:	KC RP	5: KH = 0000;
006E DTY2	:	KC NC	6: KH = 0000;
006F DTY3	:	KC	7: KH = 0000;
0070 WER1	:	KF +1	8: KH = 0000;
0071 WER2	:	KF +15	9: KH = 0000;
0072 WER3	:	KF +0	10: KH = 0000;
0073 ZFPN	:	KC B4	11:
0074 ZIEL	:	KY 150,1	
0075	:		
0076	:		
0077	:		

Example 2: Transfer from FW160, FW162 to R parameters R 50 depending on I 5.6.

0088	:		
0089	:	A I 5.6	
008A	:	JC FB 62	
008B NAME	:	NCD-SCHR	FW 160 KH = 00EF
008C SCHR	:	F 0.1	FW 162 KH = 1BA1
008D NSBY	:	DR 5	
008E ANZ	:	KF +1	
008F DTY1	:	KC RP	R50 = 156 701.77
0090 DTY2	:	KC NC	
0091 DTY3	:	KC	
0092 WER1	:	KF +1	
0093 WER2	:	KF +50	
0094 WER3	:	KF +0	
0095 ZFPN	:	KC F3	
0096 QUEL	:	KY 0,160	
0097	:		
0098	:		

6.3.4 FB 65 M STACK Transfer flags to stack FB 66 STACK M Flag stack to transfer flag area

1. Description

FB 65

FB 65 is used to save flag area FB 224 - FB 255 in the flag stack to protect intermediate results and transfer flags of the function blocks against overwriting, e. g. when using nested calls (one FB calls another FB which uses the same flag area). FB 65 can be called in conjunction with FB 66 only.

FB 66

Function block FB 66 writes flag bytes FB 224 - FB 255, which were saved in the flag stack by FB 65, back to their original locations, thus ensuring that the function block currently executing has the correct historical values at its disposal.

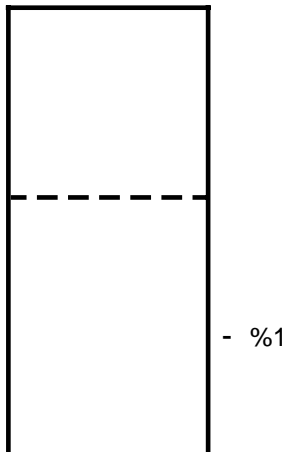
FB 66 can be called only in conjunction with FB 65.

2. Block data FB 65, FB 66

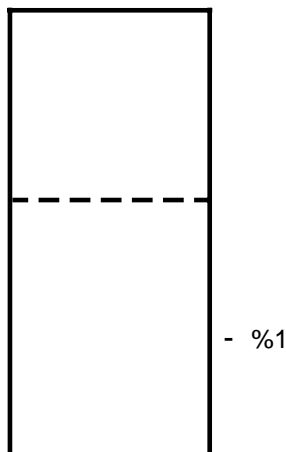
Bit no. :
FBs to be loaded : None
Type of FB call : Unconditional or conditional
DBs to be input : None
Error messages : %1 Stack pointer overflow on flag entry

3. Block call

FB 65 M STACK



FB 66 STACK M



7 Transfer Parameters and Operating System Machine Data

7.1 Transfer Parameters

Specific bits are set in flag bytes FB 0 to FB 24 in the restart routine and during program scanning.

FB	Bit: 7	6	5	4	3	2	1	0	Comments
0	Flashing frequency 1 Hz						One	Zero	Basic signal
1									Actual OB No.
2						OB 2	OB 1		Initial state
3						OB 2	OB 1	OB 20	Cold restart
4									
5									
6						OB 2			Processing delay
7									
8								Group error peripheral	Ready
Negative edge									
12	En.7	En.6	En.5	En.4	En.3	En.2	En.1	En.0	Alarm input byte
Positive edge									
16	En.7	En.6	En.5	En.4	En.3	En.2	En.1	En.0	Alarm input byte
20	S1.7	S1.6	S1.5	S1.4	S1.3	S1.2	S1.1	S1.0	Control signal byte 1 control signal byte 2
21	S2.7	S2.6	S2.5	S2.4	S2.3	S2.2	S2.1	S2.0	
22	Image fast input byte						Measuring probe activated 1 2		NC inputs
23									
24	2 nd inter- face running	1 st inter- face running			NC ready2 1)	NC ready1 2)	Battery fault	NC alarm	Monitoring

1) NC alarm with shutdown

2) NC alarm with cancellation of readin enable

n = alarm input byte (defined via PLC MD 0)

Initial state

The PLC system program provides the following basic signals:

Zero (F 0.0) : Flag with defined zero signal

One (F 0.1) : Flag with defined one signal

Flashing frequency 1Hz (F0.7) : Flashing signal with frequency 1Hz. The pulse/pause ratio is 1:1.

Current OB no.

OB no. of the OB which is currently being processed (fixed-point number)

No.	Assigned OBs
1	OB 1 (cyclic PLC user program)
2	OB 2 (alarm-controlled PLC user program)

Initial start, restart

The initial state signals/restart signals are set following a cold restart, and are reset when the cyclic user program or interrupt service routine has run a complete cycle.

Processing delay OB 2 (F 6.2)

In the interrupt service routine is reinvoked before it has terminated, this bit is set. In this case, the PLC stops. All bits are reset on cold or warm restart.

Ready state

The flag "Group error I/O" (F 8.0) is set if the PLC does not go into stop mode on a PLC error (e.g. fault in distributed I/Os, note PLC MD 2003 bit 2 (PLC stop if distributed I/Os faulted)). In this case DW 8 (diagnostics function, note PLC MD 2003 bit 7 (enable diagnostics function)) in DB 1 must be evaluated.

Alarm input byte

If a signal state change occurs at a bit of the alarm input byte (defined via PLC MD 0), a negative edge (change from state "1" to "0") sets the corresponding bit in FY 12 and a positive edge (change from state "0" to "1") sets the bit in FY 16. These flags retain their signal state until OB 2 has been processed and are then reset.

Control signal bytes 1 and 2, NC inputs

FY 20,21 and FY 22 are described in the interface reference manual SINUMERIK 805, Part 1: Signals, Section 3.3.2.

Monitoring

The appropriate flags are set if the following applies:

- NC alarm signal (F24.0)
- Battery fault signals (F24.1)
- NC ready 1 missing (F24.2)
- NC ready 2 missing (F24.3)
- 1st interface running (F24.6)
- 2nd interface running (F24.7)

7.2 PLC machine data SINUMERIK 805

System data (PLC MD 0 to 19)

MD no.	Description	Default value	Max. value	Input resolution
0	No. of the interrupt input byte	3	31	---
1	Max. percentage interpreter runtime (OB1+OB2)	15 %	20 %	1 %
2				
3	Max. interpreter runtime (OB2)	2000	2500	µs
4				
5	Cycle time monitoring	300	320	1 ms
6	No. of the last active S5 timer	15	31	---
7				
8	Interface for DB 37	1	2	---
9				
10	1st input byte DMP ¹⁾ with user add. 2	0	73	---
11	1st output byte DMP with user add. 2	0	63	---
12	1st input byte DMP with user add. 3	0	73	---
13	1st output byte DMP with user add. 3	0	63	---
14	1st input byte DMP with user add. 4	0	73	---
15	1st output byte DMP with user add. 4	0	63	---
16	1st input byte DMP with user add. 5	0	73	---
17	1st output byte DMP with user add. 5	0	63	---
18	1st input byte DMP with user add. 6	0	73	---
19	1st input byte DMP with user add. 6	0	63	---

1) DMP = Distributed machine peripherals

User MD words (PLC MD 1000 to 1007)

MD no.	User MD words
1000	FY 120 and FY 121
1001	FY 122 and FY 123
1002	FY 124 and FY 125
1003	FY 126 and FY 127
1004	FY 128 and FY 129
1005	FY 130 and FY 131
1006	FY 132 and FY 133
1007	FY 134 and FY 135

Note:

PLC MD 1000 to 1007 can be used by the user.
On every NEW START of the PLC the content of this PLC MD is transferred to FY 120 to FY 135 where they can be processed by the PLC program.

7.3 PLC machine data bits

System bits (PLC MD 2000 to 2003)

MD No.	Bit: 7	6	5	4	3	2	1	0
2000	FW 134 BCD-coded	FW 132 BCD-coded	FW 130 BCD-coded	FW 128 BCD-coded	FW 126 BCD-coded	FW 124 BCD-coded	FW 122 BCD-coded	FW 120 BCD-coded
2001	H No. BCD-coded	T No. BCD-coded	S No. BCD-coded	M No. BCD-coded	DB 37 BCD-coded (from SW 2.2)			
2002		Hand-held unit available	Rapid traverse/feed- rate override for 3rd and 4th axis (only T)		Transfer from the input image to the output image	No display of alarm 6163 (from SW 4.1)		No PLC alarm processing (OB2)
2003	Enable of the diagnostics function (stored in the DB1)	Fragmen- tation of STEP 5 program pro- cessing		Enable of the STEP 5 system commands		PLC STOP if distributed peripherals faulted	PLC STOP on runtime violation OB1+OB2	PLC STOP on runtime violation OB2

User MD bits (PLC MD 3000 to 3003)

MD no.	User MD bits
3000	FY 116
3001	FY 117
3002	FY 118
3003	FY 119

Note:

PLC MD 3000 to 3003 can be used by the user. On every NEW START of the PLC the content of this PLC MD is transferred to FY 116 -FY 119 where they can be processed by the PLC program.

8 Error Analysis

8.1 Types of error

The PLC operating system can detect internal errors, user programming errors and invalid machine data.

Errors may occur in the restart routine or during cyclic scanning. Errors may result in a PLC stop, or the user receives two kinds of error message:

- 1.) Over an error display on the NC screen in the form of a PLC interrupt
- 2.) By the setting of a group bit in the basic signal flag area (F 8.0) and of a detailed error code bit in DB1. The user can then react to the error at the software level. The PLC operating system does not automatically reset these bits on completion of a cycle, the user must reset them himself as required.

When no higher priority interrupt is pending (NC, PLC), the PLC alarm, which comprises the alarm number and an accompanying text is displayed in the message line for NC or PLC alarms. The alarm can be acknowledged and the fault message is then araised from the screen. When higher priority alarms are pending, the user can select an overview in the data area of the SINUMERIK 805 by pressing the softkeys "Diagnostics", "Message" and "PLC alarms".

Alarm No. 3 and the text "PLC Stop" are always displayed in a message line for NC and PLC alarms when the fault results in a PLC stop.

A detailed analysis while the PLC is in the stop mode is possible (see Sections 8.2 and 8.3).

DB 1

DW 0 KF = Current cycle time in ms
 DW 1 KF = Runtime OB1 + OB2 in μs^*
 DW 2 KF = Cyclic interpreter runtime OB1 in μs^*
 DW 3 KF = Alarm controlled interpreter runtime OB2 in μs^*
 DW 4 KF = Number of alarm processings occurred *
 DW 9 KF = 8000 (activation bit 9.15)

* KF only up to 32767, larger values are represented in KH! Values refer to the current cycle time.

Important: The activation bit D 9.15 is reset by the PLC operating system at the beginning of the cycle when the actual values are transferred to DB 1.

Note: The diagnostics function must be enabled with PLC MD 2003 bit 7. There are two possibilities for activating the diagnostics function (see next page):

- 1) Periodic display via the PG function "STATUS" and corresponding activation program in the OB 1
- ```

OB 1
Q DB 1 Call diagnostics DB
A F 0.7 Flashing frequency 1 Hz
=D 9.15 Activation bit for diagnostics
:
:
BE

```

The DB 1 is called and updated every second. In this way fluctuations in runtimes can be accurately estimated (cycle time and interpreter runtime are not constant quantities!). It is also only in this way that the alarm controlled interpreter runtime (OB2) and the number of alarm processings which occurred per cycle can be represented (e.g. simulate change in the alarm input byte).

- 2) Periodic display of individual quantities on the NC screen via PLC STATUS and the appropriate activation program in OB 1

```

OB 1
Q DB 1 Call diagnostics DB
A F 0.7 Flashing frequency 1 Hz
= D 9.15 Activation bit for diagnostics
L DW 0 Load cycle time (for example)
T FW 200 Transfer cycle time to FW 200
:
:
BE

```

**Note:** FW 200 contains the current cycle time in hexadecimal (FW 200 KH=...), which can also be displayed in decimal KF ... with a PG.

**Description of the diagnostics DB**

- DW 0 : Current cycle time (ms)
- DW 1 : Current interpreter runtime OB1+ OB2 (µs)
- DW 2 : Current cyclic interpreter runtime OB1 (µs)
- DW 3 : Current alarm control interpreter runtime (µs)
- DW 4 : Number of alarm processings per cycle

- DR8 bit 0 MPC does not respond (MPC node DL10)
- bit 1 Distributed transfer faulty
- bit 4 MD10...MD19 start address of distributed peripherals incorrect
- bit 5 DMP group fault

DL10 MPC node belonging to DR8

- DR 15 bit 0...bit 4 Module code associated with DMP node
  - 1 F<sub>Hex</sub> = DMP-16E/16A
  - 1 E<sub>Hex</sub> = DMP-32E
  - 1 C<sub>Hex</sub> = Hand-held unit
  - 1 7<sub>Hex</sub> = Machine control panel DMP module of the flat operator panel
- bit 6 Bit for DMP 24V supply not ok
- bit 7 Bit for overtemperature in DMP module (LOW active)

**Note:** The values measured refer to the cycle time which just occurred, and is contained in DW 0.

DW 8 to DW 15 : If the PLC does not go into stop mode on faults indeed DW 8 to DW 15, F 8.0 (group fault peripheral) is set.



## 8.2 Interrupt stack

The programmer's "Output STACK" function can be invoked to help analyze errors. This function delays the control bits. Some PG software releases display a bit identifier in the otherwise unused display locations; this identifier is of no relevance for the PLC105W. On the other hand, identifiers marked with \* (for unused) may have a special meaning for the PLC105W.

### CONTROL BITS

|                |               |               |                 |               |                |               |                |
|----------------|---------------|---------------|-----------------|---------------|----------------|---------------|----------------|
| <b>NB</b>      | <b>NB</b>     | <b>BSTSCH</b> | <b>SCHTAE</b>   | <b>ADRBAU</b> | <b>SPABBR</b>  | <b>NAUAS</b>  | <b>NB</b>      |
| <b>NB</b>      | <b>NB</b>     | <b>NNN*</b>   | <b>PERUNKL*</b> | <b>NB</b>     | <b>NB</b>      | <b>NB</b>     | <b>NB</b>      |
| <b>STOZUS</b>  | <b>STOANZ</b> | <b>NEUSTA</b> | <b>WIEDAN</b>   | <b>BATPUF</b> | <b>NB</b>      | <b>NB</b>     | <b>NB</b>      |
| <b>KEINPS*</b> | <b>UAFEHL</b> | <b>MAFEHL</b> | <b>EOVH</b>     | <b>NB</b>     | <b>NB</b>      | <b>OBWIED</b> | <b>NB</b>      |
| <b>NB</b>      | <b>NB</b>     | <b>KOPFNI</b> | <b>NB</b>       | <b>WECKFE</b> | <b>PADRFE</b>  | <b>ASPLUE</b> | <b>RAMADFE</b> |
| <b>EAADFE*</b> | <b>SYNFEH</b> | <b>NINEU</b>  | <b>NIEWIED</b>  | <b>RUFBST</b> | <b>QVZNIN</b>  | <b>SUMF</b>   | <b>URLAD</b>   |
| <b>NB</b>      | <b>NB</b>     | <b>STS*</b>   | <b>STP*</b>     | <b>TBWFEH</b> | <b>LIR/TIR</b> | <b>NB</b>     | <b>NB</b>      |
| <b>NB</b>      | <b>LPTP*</b>  | <b>NB</b>     | <b>NB</b>       | <b>NB</b>     | <b>NB</b>      | <b>NB</b>     | <b>NB</b>      |

The control bits in detail:

|                |                                                                        |
|----------------|------------------------------------------------------------------------|
| <b>BSTSCH</b>  | Coordination bit for block shift                                       |
| <b>SCHTAE</b>  | Memory compression facility active                                     |
| <b>ADRBAU</b>  | Successful address table generation following cold/warm restart        |
| <b>SPABBR</b>  | Memory compression aborted                                             |
| <b>NAUAS</b>   | Decentralized mains power failure                                      |
| <b>NNN</b>     | Illegal OP-code/invalid command parameter                              |
| <b>PERUNKL</b> | I/Os not ready (in expansion unit)                                     |
| <b>STOZUS</b>  | PLC in STOP mode (may be set/reset by OS only)                         |
| <b>STOANZ</b>  | STOP status flag                                                       |
| <b>NEUSTA</b>  | Cold restart flag                                                      |
| <b>WIEDAN</b>  | Warm restart flag                                                      |
| <b>BATPUF</b>  | Power supply unit is battery-backed                                    |
| <b>KEINPS</b>  | No user program memory or user data memory submodule                   |
| <b>UAFEHL</b>  | STOP status 15 due to interrupt event.                                 |
| <b>MAFEHL</b>  | Restart error flag (PLC could not go into normal operation)            |
| <b>EOVH</b>    | Input bytes available for process alarms (four successive inputs)      |
| <b>OBWIED</b>  | Organization block for warm restart (OB20) active                      |
| <b>KOPFNI</b>  | Block header in user memory cannot be interpreted                      |
| <b>WECKFE</b>  | Group flag for user program runtime error (OB1, OB2)                   |
| <b>PADRFE</b>  | Bad EPROM                                                              |
| <b>ASPLUE</b>  | User memory addressing not contiguous                                  |
| <b>RAMADFE</b> | Bad RAM                                                                |
| <b>EAADFE</b>  | Error in I/O area                                                      |
| <b>SYNFEH</b>  | Invalid block length in user memory or bad block synchronization word  |
| <b>NINEU</b>   | No cold restart possible (bootstrapping required)                      |
| <b>NIEWIED</b> | No warm restart possible (cold restart required)                       |
| <b>RUFBST</b>  | Non-existent block called                                              |
| <b>QVZNIN</b>  | Timeout                                                                |
| <b>SUMF</b>    | Sumcheck error                                                         |
| <b>URLAD</b>   | Overall reset and subsequent bootstrap loading of user memory required |
| <b>STS</b>     | Direct STOP initiated via STS                                          |
| <b>STP</b>     | Interrupt condition code following STP                                 |
| <b>TBWFEH</b>  | Timeout, block transfer operation                                      |
| <b>LIRTIR</b>  | Timeout, LIR or TIR operation                                          |
| <b>LPTP</b>    | Timeout, direct access to I/Os via user statements                     |

The following can be called to screen as the next display:

**INTERRUPT STACK**

```

DEPTH: 01
BEF-REG: 0000 SAZ: 0000 DB-ADR: 0000 BA-ADR: 000
BST-STP: 0000 XX-NR.r: DB-NR.: YY-NR.:
REL-SAZ: DBL-REG:
VEK-ADR: 0000 UAMK: 0000 UALW: 0000
ACCU1: 0000 0000 ACCU2: 0000 0000 ACCU3: 0000 0000 ACCU4: 0000 0000
RESULT COND. CODES: CC1 CCO OVFL OVFLS ODER STATUS VKE ERAB
CAUSE OF
FAULT: STOPS STUEB NAU QVZ ZYK BAU SUF STUEU ADF PARI TRAF

```

Bits ACCU3 and ACCU4 are irrelevant to the PLC, as it is not equipped with these accumulators. Like the control bits, the bits marked with "-" may show an irrelevant bit identifier. Refer to Section 9.2.2 for details on the result condition codes.

- DEPTH**      The latest interrupt event (DEPTH = 01) is always shown for the PLC
- BEF-REG**    Operation representation (hexadecimal)
- SAZ**        Step address counter  
               The operation representation is ascertained by accessing user program memory via the step address counter. Since the step address counter always points to the next operation to be executed, the operation that is interpreted is the one located at the next lower address. The information shown in the BEF-REG is thus incorrect when double-word operations are involved as well as immediately following jump commands.
- DB-ADR**     Block start address
- BST-STP**    Block stack pointer
- XX-NR.**     Current block whose execution was initiated by the interrupt
- DB-NR.**     Current data block
- YY-NR.**     Block that called the current block (XX)
- REL-SAZ**    Relative step address counter in the current block, incorrect statement is located at the previous address
- DBL-REG**    Register containing the data block length
- VEK-ADR**    Vector address for external storage
- UAMK**       Control register 1 (hexadecimal) (see next page)
- UALW**       Control register 2 (non-existent)
- STOPS**      Irrelevant
- STUEB**     Block stack overflow (max. 12 entries allowed)
- NAU**        Mains power failure (controller remains in the STOP mode until power is recovered)
- QVZ**        Timeout
- ZYK**        Cycle scan time exceeded
- BAU**        Battery failure
- SUF**        Substitution error
- STUEU**     Interrupt stack overflow (max. 2 levels possible)
- ADF**        Addressing error
- PARI**       Parity error
- TRAF**       Transfer error

## 8.3 Detailed error code

### 8.3.1 Display on programmer

Using the programmer's info function, the user can display additional information for interrupt analysis by entering pseudo address F000<sub>hex</sub>. The following is displayed on the programmer screen:

| ADR  | VAL  | ADR  | WERT | ADR  | WERT | ADR  | WERT |
|------|------|------|------|------|------|------|------|
| F000 | 00xx | F001 | 61yy | F002 | xxx1 | F003 | xxx2 |
| F004 | xxx3 | F005 | zob2 | F006 | 0000 | F007 | 0000 |
| F008 | 0000 | F009 | 0000 | F00A | 0000 | F00B | 0000 |
| F00C | 0000 | F00D | 0000 | F00E | 0000 | F00F | 0000 |
| F010 | 0000 | F011 | 0000 | F012 | 0000 | F013 | 0000 |
| F014 | 0000 | F015 | 0000 | F016 | 0000 | .... |      |

Addresses F00C to F011 are described in detail in Section 9.1.2; subsequent displays are irrelevant.

The following applies to addresses F000 to F005:

**xx = Internal detailed error code (ERRCODE for PLC STOP)**

**61yy = NC alarm number 6100 - 6163**

**xxx1 = Auxiliary error info, word 1**

**xxx2 = Auxiliary error info, word 2**

**xxx3 = Auxiliary error info, word 3**

**zob2 = Event counter, processing timeout in OB 2**

The auxiliary error info enables more precise analysis of the reason for a timeout or parameter initialization error. All auxiliary error info is deleted on a cold restart.

#### 1. PLC-QVZ:

The OP-code of the command that caused the timeout errors is entered in word 1 (xxx1). xxx2/xxx3 contain the following:

- Timeout caused by LIR, TIR, TNB or TNW:
  - xxx2**      Offset address
  - xxx3**      Segment number  
of the non-addressed memory
- Timeout caused by substitution operations:
  - xxx2**      Substitution operation
- Timeout caused by LPB, LPW, TPB, TPW operations:
  - xxx2**      = 000E (Timeout during loading of the input modules)
  - = 000A (Timeout during transport to the output modules)
  - xxx3**      = Byte address (BCD) of the operation parameter

#### 2. Parameter initialization errors

**xxx3**      = Number (BCD) of the invalid operation parameter (peripheral byte no., segment no., block no., timer/counter no.)

The "Event counter, timeout" shows how often the interrupt service routine (called over OB2) was requested before it could terminate its original run.

The number of the organization block which caused the delay is entered in word 1 (xxx1) of the auxiliary error info.

Internal detailed error code xx differs from NC alarm number 61yy as follows:

- xx - Hexadecimal value between 100 and 163 (64h - 13h)
  - Value 00 always results in a PLC STOP
- 61yy - BCD value between 6100 and 6163 (on NC screen also)
  - A value 0000 does not always result in a PLC STOP

The table below provides an overview of the error code allocated to an error event for both the xx and 61yy identifiers:

| Error identifier / Error event                         | XX                                                 | 61yy       |
|--------------------------------------------------------|----------------------------------------------------|------------|
| Error detected during scanning of the STEP 5 program   | 100-115<br>(64 <sub>hex</sub> -73 <sub>hex</sub> ) | 6100-6115  |
| Error detected during startup                          | 116-143<br>(74 <sub>hex</sub> -8F <sub>hex</sub> ) | 6116 -6143 |
| Error detected during scanning of the cyclic program   | 144-149<br>(90 <sub>hex</sub> -95 <sub>hex</sub> ) | 6144-6149  |
| Error messages output by the interrupt service routine | 150-163<br>(96 <sub>hex</sub> -A3 <sub>hex</sub> ) | 6150-6163  |

### 8.3.2 Display on screen

When the PLC is in the stop state alarm 3 (PLC stop) appears on the screen, as long as no alarm with higher priority is pending. You can display the cause of the PLC stop state (PLC alarms 6100 to 6163) in the operator interface if you press softkey DIAGNOSIS and the PLC ALARM. You can also display PLC user alarms (alarms 6000 to 6063) in this overview. After having remedied the error, you can acknowledge PLC alarms 3 and 6100 to 6163 only with a POWER-ON of the control or with a programmer using the PLC START function.

**8.4 Alarm list**
**SYSTEM 805 ALARM LIST  
PLC OPERATING SYSTEM AND USER ALARMS**

| Alarm no. | Description     | PLC response |
|-----------|-----------------|--------------|
| 6000      | PLC user alarms | Message      |
| .         | ”               | ”            |
| 6063      | PLC user alarms | Message      |

| Alarm no. | Description                    | PLC response |
|-----------|--------------------------------|--------------|
| 6100      | No process interface module    | STOP         |
| 6101      | Invalid MC5 code               | STOP         |
| 6102      | Invalid MC5 parameter          | STOP         |
| 6103      | Transfer to non-existent DB    | STOP         |
| 6104      | Substitution error             | STOP         |
| 6105      | No MC5 block                   | STOP         |
| 6106      | No DB                          | STOP         |
| 6107      | Invalid segment LIR/TIR        | STOP         |
| 6108      | Invalid segment block transfer | STOP         |
| 6109      | Block stack overflow           | STOP         |
| 6110      | Interrupt stack overflow       | STOP         |
| 6111      | MC5 command STS                | STOP         |
| 6112      | MC5 command STP                | STOP         |
| 6113      | Invalid MC5 timer/counter      | STOP         |
| 6114      | Function macro                 | STOP         |
| 6115      | System operations disabled     | STOP         |
| 6116      | MD 0:Interrupt input           | STOP         |
| 6117      | MD 1: CPU load                 | STOP         |
| 6118      | MD 3: Alarm rout. runtime      | STOP         |
| 6119      | MD 5: Cycle scan time          | STOP         |
| 6120      |                                |              |
| 6121      | MD 6: Last MC5 timer           | STOP         |

|                                                                             |  |  |
|-----------------------------------------------------------------------------|--|--|
| <b>SYSTEM 805 ALARM LIST</b><br><b>PLC OPERATING SYSTEM AND USER ALARMS</b> |  |  |
|-----------------------------------------------------------------------------|--|--|

| Alarm no. | Meaning                                  | PLC response |
|-----------|------------------------------------------|--------------|
| 6122      |                                          |              |
| 6123      | Invalid sampling time                    | STOP         |
| 6124      | Gap in MC5 memory                        | STOP         |
| 6125      | Multiple input definitions               | STOP         |
| 6126      | Multiple output definitions              | STOP         |
| 6127      | No alarm byte                            | STOP         |
| 6128      |                                          |              |
| 6129      |                                          |              |
| 6130      | Sync. error, basic prog.                 | STOP         |
| 6131      | Sync. error, MC5 prog.                   | STOP         |
| 6132      | Sync. error, MC5 data                    | STOP         |
| 6133      | Invalid block, basic prog.               | STOP         |
| 6134      | Invalid block, MC5 prog.                 | STOP         |
| 6135      | Invalid block, MC5 data                  | STOP         |
| 6136      | Sumcheck error in MC5 block              | STOP         |
| 6137      | Sumcheck error in basic prog.            | STOP         |
| 6138      | No reaction from MPC mode                | *            |
| 6139      | MPC transfer error                       | *            |
| 6140      | MD10-MD19 I/O<br>start address incorrect | STOP         |
| 6141      |                                          |              |
| 6142      |                                          |              |
| 6143      |                                          |              |
| 6144      |                                          |              |
| 6145      |                                          |              |
| 6146      |                                          |              |
| 6147      | Mod. in global I/Os                      | *            |

|                                                                             |
|-----------------------------------------------------------------------------|
| <b>SYSTEM 805 ALARM LIST</b><br><b>PLC OPERATING SYSTEM AND USER ALARMS</b> |
|-----------------------------------------------------------------------------|

| Alarm-no. | Meaning                                   | PLC response |
|-----------|-------------------------------------------|--------------|
| 6148      | Temperature rise in expansion unit        |              |
| 6149      | STOP over PG softkey                      | STOP         |
| 6150      | Timeout: User memory                      | STOP         |
| 6151      | Timeout: Link memory                      | STOP         |
| 6152      | Timeout: LIR / TIR                        | STOP         |
| 6153      | Timeout: TNB / TNW                        | STOP         |
| 6154      | Timeout: LPB / LPW / TPB / TPW            | STOP         |
| 6155      | Timeout: Substitution command             | STOP         |
| 6156      | Timeout cannot be interpreted             | STOP         |
| 6157      | Timeout: JU FB/JC FB                      | STOP         |
| 6158      | Timeout on I/O transfer                   | STOP         |
| 6159      | STEP 5 prog. runtime exceeded             | *            |
| 6160      | OB2 runtime exceeded                      | *            |
| 6161      | Cycle scan time exceeded                  | STOP         |
| 6162      | Delay OB2                                 | *            |
| 6163      | Time monitoring PLC network communication |              |
| 6164      | No reaction from MPC mode                 |              |
| 6165      | DMP logic supply (24V) not o.k.           |              |
| 6166      | DMP overtemperature (>63°)                |              |

| Alarm no. | Description                   | PLC response |
|-----------|-------------------------------|--------------|
| 7000      | PLC user operational messages | Message      |
| .         | "                             | "            |
| 7063      | PLC user operational messages | Message      |

\* PLC response depends on MD definition (STOP or message only)

## 9 STEP 5 command set with programming examples

### 9.1 Memory organization

In order to be able to reach address areas outside the normal range for STEP 5 commands (e.g. process image) with the "Address info" function of a programmer (such as PG675, PG685), the so-called "segment switch" is needed. This is an auxiliary variable with which a particular segment within the 1 Megabyte address area of the processor can be reached.

Each address then comprises the offset address (position in the segment) and the segment address (depending on the position of the segment switch).

The following segments can be selected by the user via a segment number either with the "Address info" diagnostic function of a programmer or with special Step 5 system commands (c.f. Section 9.4.13):

| Segment number | Designation | Segment address     | Offset address (word-oriented)           | Remarks                                                                                 |
|----------------|-------------|---------------------|------------------------------------------|-----------------------------------------------------------------------------------------|
| 1              | PSEG        | 2000 <sub>hex</sub> | 0000 <sub>hex</sub> -15FF <sub>hex</sub> | Read and write                                                                          |
| 2              | NCPERSEG    | 4000 <sub>hex</sub> | 0000 <sub>hex</sub> -3000 <sub>hex</sub> | Read and write from 2000 <sub>hex</sub> on, otherwise possible PLC timeout and PLC stop |
| 3              | BSEG        |                     | 0000 <sub>hex</sub> -801 <sub>hex</sub>  | Read and write                                                                          |
| 4              | VSKSEG      |                     | 0000 <sub>hex</sub> -230 <sub>hex</sub>  | Read and write                                                                          |
| 5              | AWPSEG      | 1000 <sub>hex</sub> | 0000 <sub>hex</sub> -17FF <sub>hex</sub> | Read and write                                                                          |
| 6              | AWDSEG      | 1300 <sub>hex</sub> | 0000 <sub>hex</sub> -C9F <sub>hex</sub>  | Read and write                                                                          |
| 7              | SY1SEG      |                     | 0000 <sub>hex</sub> -7FFF <sub>hex</sub> | Read                                                                                    |
| 8              | SY2SEG      |                     | 0000 <sub>hex</sub> -2000 <sub>hex</sub> | Read                                                                                    |
| 9              | PPSEG       |                     | No PG info function                      | Read                                                                                    |
| A              | PPWFSEG     |                     | No PG info function                      | Read                                                                                    |

If another value is defaulted for the segment switch or segment number, it is ignored (with the programmer Info function).

On cold restart position 5 of the segment switch is automatically selected; in this range, there is the AWPSEG user program segment.



**Memory organization:**

Segment  
number

|   |     |                            |                                                                                                                               |
|---|-----|----------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| 1 | *   | PLC system data:           | Process images, block lists, memory administration, BSTACK, ISTACK etc. ( )                                                   |
| 2 |     | Hardware components:       | I/O interface module, processor register, MPC link memory for global I/Os ( )                                                 |
| 3 | *   | Internal NC-PLC interface: | Axis-specific signals channel-specific signals NC operator panel signals etc.                                                 |
| 4 | *   | Spindle interface:         | Spindle-specific signals                                                                                                      |
| 5 | **  | User data memory:          | Max. 8 k bytes (16 kbytes from SW 4.1) for Step 5 user program (OB, PB, SB, FB)                                               |
| 6 | **  | User data memory:          | Max. 4 k bytes for Step 5 user data blocks and 0.3 k bytes for data blocks initialized by the PLC operating system on startup |
| 7 | *** | PLC system program:        | Programming and test functions, interpreter, etc. ( 50 kB)                                                                    |
| 8 | *** | Resident function macros:  | FB11, FB60 etc. ( 16k bytes)                                                                                                  |
| 9 |     | Part program memory        |                                                                                                                               |
| A |     | Reserved                   |                                                                                                                               |

\* Internal RAM area on the main board  
 \*\* Buffered RAM area on the main board  
 \*\*\* Available in the system memory in the EPROM

### 9.1.1 Changing the segment switch

The following sequence applies to the PG675 programmer:

- Call the information functions with the F7 key (information)
- Call any memory areas with the F8 key (output address)
- Enter the pseudo-address E000<sub>hex</sub>
- Press the Enter key and then
- Press the Abort key immediately

The first word shown is the current setting of the segment switch.

- Press the correction key
- Enter the new segment address

### 9.1.2 Block lists

The start addresses of the block lists can be output by entering pseudo-address F000 in segment 1 (see Section 8.3 for the significance of addresses F000 to F009):

**Note:** PG675/685 accepts address F000 as the correct input.

|             |                                    |
|-------------|------------------------------------|
| <b>F000</b> | see Section 8.3                    |
| :           | :                                  |
| <b>F009</b> | see Section 8.3                    |
| <b>F00C</b> | Start address of the OB block list |
| <b>F00D</b> | Start address of the FB block list |
| <b>F00E</b> | Start address of the DB block list |
| <b>F010</b> | Start address of the SB block list |
| <b>F011</b> | Start address of the PB block list |

These start addresses contain the address of the first block of the block list, that is OB0 (or FB0, DB0, SB0 or PB0).

Two words per block are needed in the block list.

The first word is the offset address, the second word is the segment address.

In these words the high byte and the low byte are interchanged. It is therefore necessary to swap them back again.

For the DBs the word-oriented offset addresses must be divided by two after the high/low swap to obtain the correct address for the byte-oriented data blocks.

OBs, PBs, FBs and SBs are located in segment 5 (address 1000<sub>hex</sub>).  
DBs are located in segment 1 (address 1300<sub>hex</sub>).

**Note:** If the blocks are output after the directory function of the programmer, blocks which are not located in the user program segment appear with an offset address increased by 8000 (applies data blocks and resident function macros).

In the following examples:                    Input and address selections into the PG675/685 are printed in *italics*.

**Example:** FB1

Select segment 1 (Section 9.1.1)

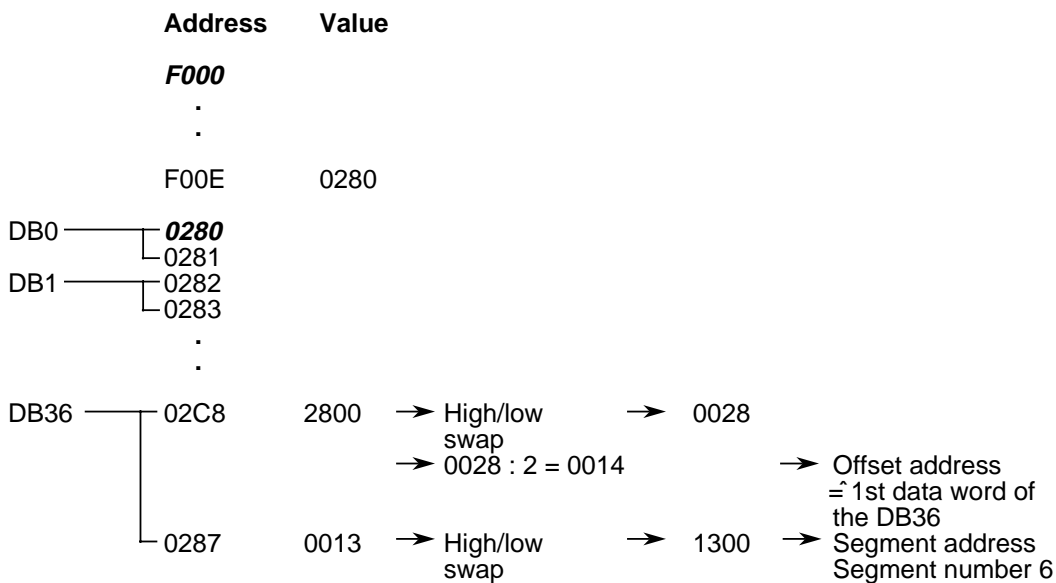
| <u>Address</u> | <u>Value</u> |      |                 |        |                                              |
|----------------|--------------|------|-----------------|--------|----------------------------------------------|
| <b>F000</b>    |              |      |                 |        |                                              |
| :              |              |      |                 |        |                                              |
| F00D           | 0080         |      |                 |        |                                              |
| FB0 ———        | 0080         |      |                 |        |                                              |
|                | 0081         |      |                 |        |                                              |
| FB1 ———        | 0082         | AB01 | → High/low swap | → 01AB | → Offset address =1st instruction of the FB1 |
|                | 0083         | 0010 | → High/low swap | → 1000 | → Segment address Segment number 5           |
| FB2 ———        | 0084         |      |                 |        |                                              |
|                | 0085         |      |                 |        |                                              |

Select segment 5 (Section 9.1.1)

| <u>Address</u> | <u>Value</u> |       |                                 |
|----------------|--------------|-------|---------------------------------|
| <b>01A4</b>    | 0000         |       |                                 |
| 01A5           | 0000         |       |                                 |
| 01A6           | 7070         | } ——— | Block header (5 addresses long) |
| 01A7           | 4801         |       |                                 |
| 01A8           | C400         |       |                                 |
| 01A9           | 0000         |       |                                 |
| 01AA           | 0111         |       |                                 |
| 01AB           | 2D05         | } ——— | 1st instruction of FB1          |
| :              | :            |       |                                 |

**Example:** DB36

Select segment 1 (Section 9.1.1)



Select segment 6 (Section 9.1.1)

| Address     | Value |                                        |
|-------------|-------|----------------------------------------|
| <b>000B</b> | 0000  |                                        |
| 000C        | 0000  |                                        |
| 000D        | 0000  |                                        |
| 000E        | 0000  |                                        |
| 000F        | 7070  | } Block header DB36 (5 addresses long) |
| 0010        | C124  |                                        |
| 0011        | C000  |                                        |
| 0012        | 0000  |                                        |
| 0013        | 0026  |                                        |
| 0014        | 0010  | } 1st data word of DB 36               |
| :           | :     |                                        |
| :           | :     |                                        |





## 9.2.2 Condition codes of the PLC

There are commands for processing individual bit information, and there are commands for processing word information (8, 16 or 32 bits). In both groups, there are commands which set condition codes and commands which interpret condition codes. For both command groups there are "condition codes for bit operations" and "condition codes for word operations". The CC byte for the PLC is as follows:

|                |      |    |                        |      |    |     |                          |  |  |  |                |
|----------------|------|----|------------------------|------|----|-----|--------------------------|--|--|--|----------------|
| 2 <sup>7</sup> |      |    | CC for word operations |      |    |     | CC for bit operations    |  |  |  | 2 <sup>0</sup> |
| CC 1           | CC 0 | OV | OS                     | STAT | OR | RLO | $\overline{\text{ERAB}}$ |  |  |  |                |

≙ status display during status operation at PG.

Condition codes for bit operations:

**$\overline{\text{ERAB}}$ :**  $\overline{\text{ERAB}}$  signifies first interrogation. This is the start of a logic operation. ERAB is set at the end of a logic operation sequence (memory operations).

**RLO:** Result of logic operation; result of bit-wide operations. Logical value for comparison commands.

**OR:** This informs the processor that the following AND operations must be handled before an OR operation (AND before OR)

**STAT:** Signal state of operand.

Condition codes for word operations:

**OV:** OVER; this indicates whether, for the arithmetic operation just terminated, the valid numeric range has been exceeded.

**OS:** OVER LATCHING; the over-bit is stored; in the course of two or more arithmetic operations, this serves to indicate whether an overflow error (OVER) has occurred at some time.

**CC1, CC0** are condition codes whose interpretation can be found in the following table.

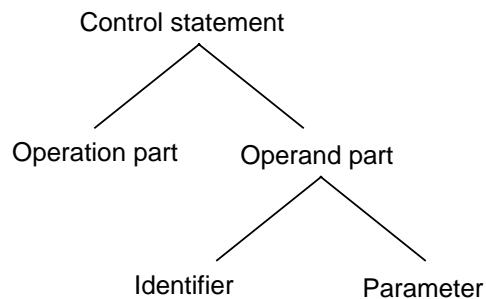
| Condition codes for word operations |      |      | Fixed-point calculation Result | Boolean result | Comparison Contents of Acc 1 + Accu 2 | Shift shifted bit | Floating-point calculation Result |
|-------------------------------------|------|------|--------------------------------|----------------|---------------------------------------|-------------------|-----------------------------------|
| CC 1                                | CC 2 | OVER |                                |                |                                       |                   |                                   |
| 0                                   | 0    | 0    | Result = 0                     | = 0            | Accu 2 = Accu                         | 0                 | Mantissa=0; Exp. allowed          |
| 0                                   | 1    | 0    | Result < 0                     | -              | Accu 2 < Accu                         | -                 | Mantissa <0; Exp. allowed         |
| 1                                   | 0    | 0    | Result > 0                     | = 0            | Accu 2 > Accu                         | 1                 | Mantissa >0; Exp. allowed         |
| 0                                   | 0    | 1    | "Over-Zero"*)                  | -              | -                                     | -                 | Mantissa =0; Exp. allowed - 128   |
| 0                                   | 1    | 1    | 0 from pos.range               | -              | -                                     | -                 | Mantissa <0; Exp. allowed + 127   |
| 1                                   | 0    | 1    | 0 from neg.range               | -              | -                                     | -                 | Mantissa >0; Exp. allowed + 127   |
| 1                                   | 1    | 1    | Division by zero               | -              | -                                     | -                 | Division by zero                  |

\*) Special case: Greatest negative number added to itself

Jump operations are available for immediate interpretation of the condition codes (see "Supplementary operations").

### 9.2.3 Step 5 command representation

A statement is structured as follows:



The operation part describes the function to be executed. It defines what the processor is to do.

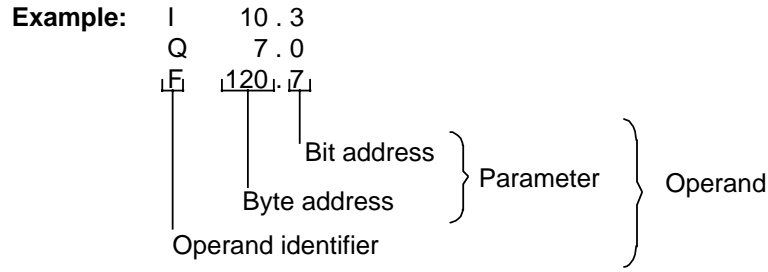
The operand part contains the information necessary for the operation to be executed. It specifies what the processor is to do something with. The Step 5 programming language has the following operand areas:

|                       |                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------|
| Inputs I              | These constitute the interface from the process to the programmable controller (process image), |
| Outputs Q             | These constitute the interface from the programmable controller to the process (process image), |
| Flags F               | These are for storing binary intermediate results,                                              |
| Data D                | These are for storing digital intermediate results,                                             |
| Timers T              | These implement timer functions,                                                                |
| Counters C            | These implement counting functions,                                                             |
| Peripherals P, Q      | The process I/Os (input/output modules) are addressed directly,                                 |
| Constants K           | Represent a fixed predefined number,                                                            |
| Blocks<br>OB,PB,FB,DB | Serve to structure the program.                                                                 |

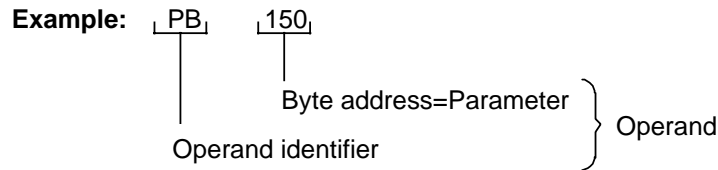
The designation of the operand areas is the (*Operand*) *identifier*. The parameter must be specified to address a certain operand in an operand area.

The *parameter* specifies the address of an operand. The operand areas inputs I, outputs Q and flags F are addressed byte-by-byte, i.e. the specified number refers to a specific byte of these operand areas (byte address). These operand areas can also be addressed bit-by-bit. The bit address is separated from the byte address with a fullstop.

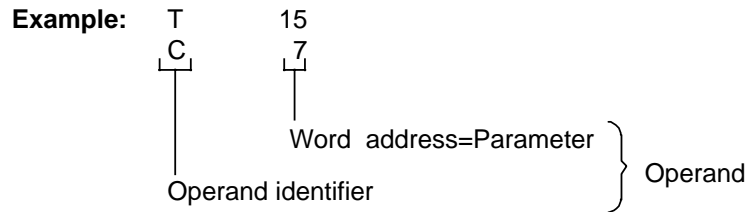




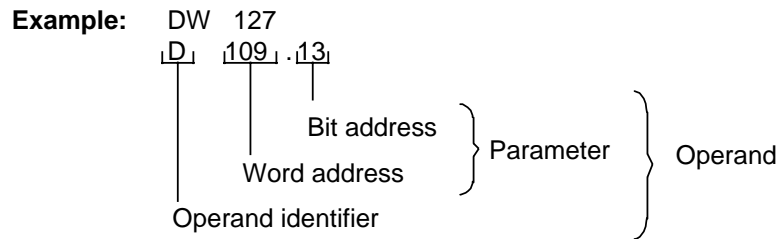
The peripherals P operand area is also addressed byte-by-byte but has no bit address.



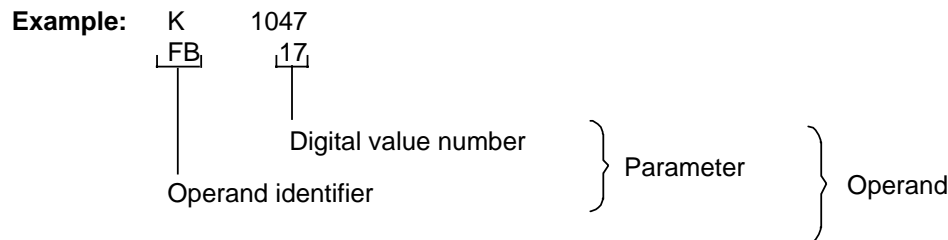
The timers T and counters C operand areas are addressed word-by-word (word address). These operand areas have no bit address.



The data D operand area can be addressed word-by-word and bit-by-bit .



The parameter of the constant K operand area represents the number which is to be processed as a digital value. The parameters of the OB, PB, FB and DB operand areas specify the number of the operands in this area.



## 9.3 Basic operations

Basic operations are programmable in program, sequence, organization and function blocks. They can be input and output in program, sequence and organization blocks in the three methods of representation (LAD, CSF and STL).

### Exceptions:

1. Load, transfer and code operations. These can only be programmed graphically, indirectly and with limits in conjunction with timing and counting operations.
2. Arithmetic operations and the Stop command (STP) can only be programmed in a statement list.

### 9.3.1 Logic operations, binary

| Operation                                                                                                                                                                                 | Parameter                                                                                                                                                                      | Function                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| )<br>A(<br>O(<br>O                                                                                                                                                                        |                                                                                                                                                                                | Right parenthesis<br>AND operation with parenthesized expressions<br>OR operation with parenthesized expressions<br>OR operation on AND functions                                                                                                                                                                                                                                                                                                                    |
| A <input type="checkbox"/> <input type="checkbox"/><br>O <input type="checkbox"/> <input type="checkbox"/><br>↑ ↑<br>I<br>Q<br>F<br>D<br>N I<br>N Q<br>N F<br>N D<br>T<br>N T<br>C<br>N C | 0.0 to 127.7<br>0.0 to 127.7<br>0.0 to 255.7<br>0.0 to 255.15<br>0.0 to 127.7<br>0.0 to 127.7<br>0.0 to 255.7<br>0.0 to 255.15<br>1 to 127<br>1 to 127<br>1 to 127<br>1 to 127 | AND operation with<br>OR operation with<br>Scanning an input for logic "1"<br>Scanning an output for logic "1"<br>Scanning a flag for logic "1"<br>Scanning data for logic "1"<br>Scanning an input for logic "0"<br>Scanning an output for logic "0"<br>Scanning a flag for logic "0"<br>Scanning data for logic "0"<br>Scanning a timer for logic "1"<br>Scanning a timer for logic "0"<br>Scanning a counter for contents >0<br>Scanning a counter for contents=0 |

Binary logic operations produce the "RLO" (result of the logic operation)

At the beginning of a logic sequence, the result depends only on the type of operation (A=AND, AN=AND NOT, O=OR, ON=OR NOT) and the queried logic level. Within a logic sequence, the RLO is formed from the type of operation, previous RLO and scanned signal state. A logic sequence is terminated by a limited-step command (e.g. storage operations).

### AND operation

| Given circuit | STEP 5 representation                                            |                |                          |
|---------------|------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                   | Ladder diagram | Control system flowchart |
|               | <pre> A   1.1 A   1.3 A   1.7 = Q 3.5                     </pre> |                |                          |

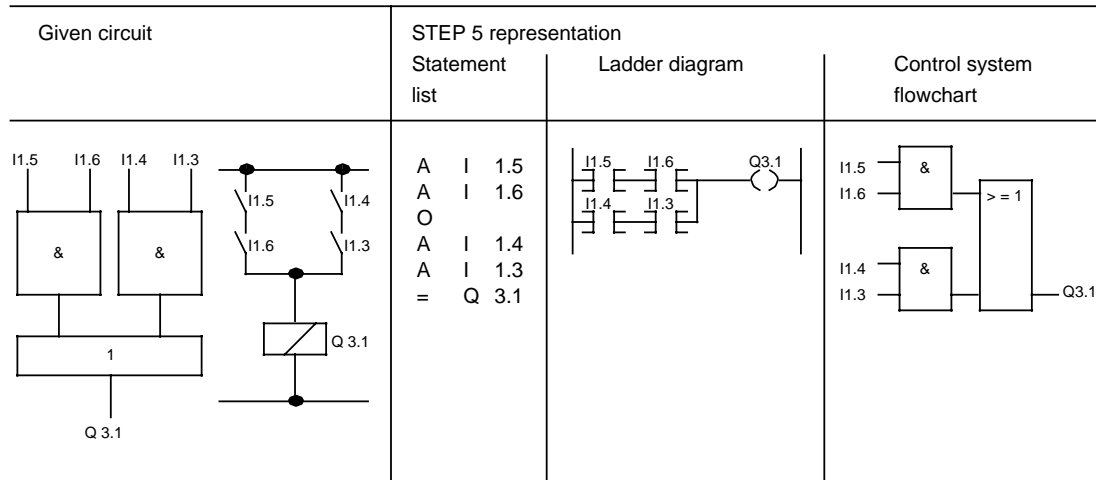
A logic 1 appears at output Q 3.5 if all inputs are simultaneously at logic 1. A logic 0 appears at output Q 3.5 if at least one of the inputs is at logic 0. The number of scans and the order of programming are arbitrary.

### OR operation

| Given circuit | STEP 5 representation                                            |                |                          |
|---------------|------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                   | Ladder diagram | Control system flowchart |
|               | <pre> O   1.2 O   1.7 O   1.5 = Q 3.2                     </pre> |                |                          |

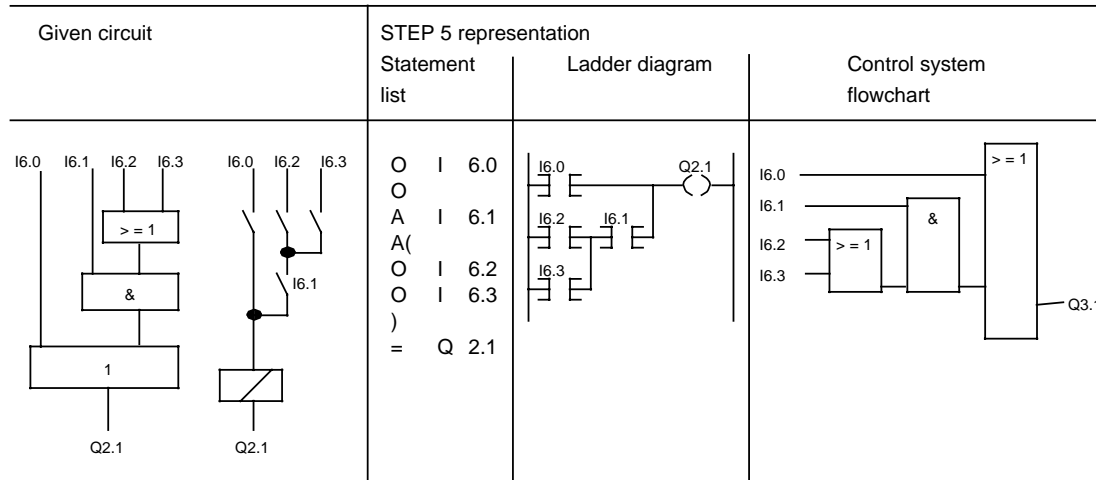
A logic 1 appears at output Q 3.2 if at least one of the inputs is at logic 1. A logic 0 appears at output Q 3.2 if all inputs are simultaneously at logic 0. The number of scans and the order of programming are arbitrary.

**AND before OR operation**



A logic 1 appears at output Q 3.1 if at least one AND condition is fulfilled. A logic 0 appears at output Q 3.1 if no AND condition is fulfilled.

**OR before AND operation**



A logic 1 appears at output Q 2.1 if input I 6.0 or input I 6.1 and one of inputs I 6.2 and I 6.3 at logic 1.  
A logic 0 appears at output Q 2.1 if input I 6.0 is at logic 0 and the AND condition is not fulfilled.

### OR before AND operation

| Given circuit | STEP 5 representation                                                              |                |                          |
|---------------|------------------------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                                     | Ladder diagram | Control system flowchart |
|               | <pre> A( O I 1.4 O I 1.5 ) A( O I 2.0 O I 2.1 ) = Q 3.0                     </pre> |                |                          |

A logic 1 appears at output Q 3.0 if both OR conditions are fulfilled. A logic 0 appears at output Q 3.0 if at least one OR condition is not fulfilled.

### Scanning for logic 0

| Given circuit | STEP 5 representation                                     |                |                          |
|---------------|-----------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                            | Ladder diagram | Control system flowchart |
|               | <pre> A I 1.5 AN I 1.6 = Q 3.0                     </pre> |                |                          |

A logic 1 only appears at output Q 3.0 if input I 1.5 is at logic 1 (N/O contact closed) and input I 1.6 is at logic 0 (N/C contact opened).

## 9.3.2 Storage operations

| Operation | Parameter                           | Function               |
|-----------|-------------------------------------|------------------------|
| S         | <input type="checkbox"/>            | Set<br>Reset<br>Assign |
| R         | <input type="checkbox"/>            |                        |
| =         | <input type="checkbox"/>            |                        |
|           | <input checked="" type="checkbox"/> |                        |
| I         | 0.0 to 127.7                        | an input               |
| Q         | 0.0 to 127.7                        | an output              |
| F         | 0.0 to 255.7                        | a flag                 |
| D         | 0.0 to 255.15                       | a data word            |

**RS Flipflop for latching signal output**

| Given circuit | STEP 5 representation                        |                |                          |
|---------------|----------------------------------------------|----------------|--------------------------|
|               | Statement list                               | Ladder diagram | Control system flowchart |
|               | <pre> A I 2.7 S Q 3.5 A I 1.4 R Q 3.5 </pre> |                |                          |

A logic 1 at input I 2.7 causes the flipflop to be set. If the logic level at input I 2.7 changes to 0, this state is retained, i.e. the signal is stored.

A logic 1 at input I 1.4 causes the flipflop to be reset. If the logic level at input I 1.4 changes to 0, this state is retained. If the set signal (input I 2.7) and reset signal (input I 1.4) are applied simultaneously, the last programmed scan (AI 1.4 in this case) is effective during processing of the rest of the program.

**RS Flipflop with flags**

| Given circuit | STEP 5 representation                        |                |                          |
|---------------|----------------------------------------------|----------------|--------------------------|
|               | Statement list                               | Ladder diagram | Control system flowchart |
|               | <pre> A I 2.6 S F 1.7 A I 1.3 R F 1.7 </pre> |                |                          |

A logic 1 at input I 2.6 causes the flipflop to be set. If the logic level at input I 2.6 changes to 0, this state is retained, i.e. the signal is stored.

A logic 1 at input I 1.3 causes the flipflop to be reset. If the logic level at input I 1.3 changes to 0, this state is retained.

If the set signal (input I 2.6) and reset signal (input I 1.3) are applied simultaneously, the last programmed scan (AI 1.3 in this case) is effective during processing of the rest of the program.



### 9.3.3 Load and transfer operations

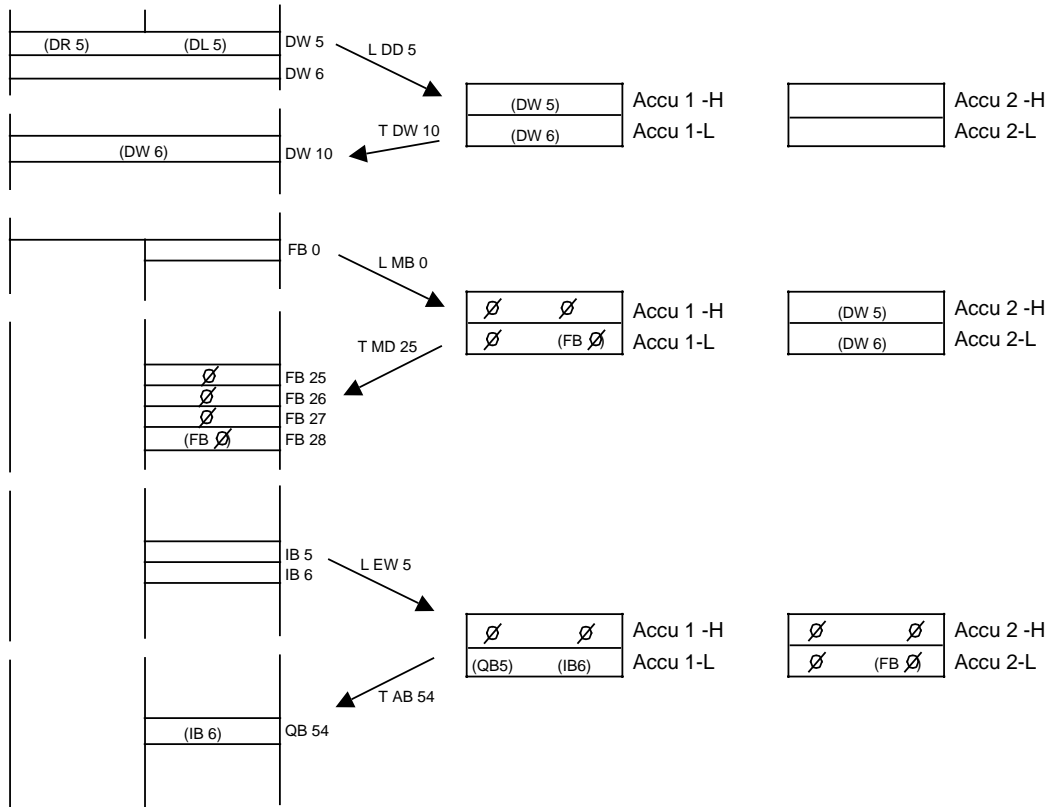
| Operation     | Parameter                                          | Functions                                 |
|---------------|----------------------------------------------------|-------------------------------------------|
| <b>L</b> □ □  |                                                    | Load                                      |
| <b>T</b> □ □  |                                                    | Transfer                                  |
| ↑ ↑           |                                                    |                                           |
| <b>I B</b>    | 0 to 127                                           | an input byte from PII 2)                 |
| <b>I W</b>    | 0 to 126                                           | an input word from the PII                |
| <b>I D</b>    | 0 to 124                                           | an input double word from the PII         |
| <b>Q B</b>    | 0 to 127                                           | an output byte from the PIO 3)            |
| <b>Q W</b>    | 0 to 126                                           | an output word from the PIO               |
| <b>Q D</b>    | 0 to 124                                           | an output double word from the PIO        |
| <b>F B</b>    | 0 to 225                                           | a flag byte                               |
| <b>F W</b>    | 0 to 254                                           | a flag word                               |
| <b>F D</b>    | 0 to 252                                           | a flag double word                        |
| <b>D R</b>    | 0 to 255                                           | an item of data (right byte)              |
| <b>D L</b>    | 0 to 255                                           | an item of data (left byte)               |
| <b>D W</b>    | 0 to 255                                           | a data word                               |
| <b>D D</b>    | 0 to 254                                           | a data double word                        |
| <b>T 2)</b>   |                                                    | 0 to 127 a time (binary)                  |
| <b>C 2)</b>   |                                                    | 0 to 127 a count (binary)                 |
| <b>P B</b>    | 0 to 255                                           | an I/O byte of the digital inputs/outputs |
| <b>P W 4)</b> | 0 to 126                                           | an I/O word of the digital inputs/outputs |
| <b>K M 1)</b> | 16-bit pattern                                     | a constant as bit pattern                 |
| <b>K H 1)</b> | 0 to FFFFH                                         | a constant in hex code                    |
| <b>K F</b>    | 0 to + (216-1)                                     | a constant as fixed point number          |
| <b>K Y 1)</b> | 0 to 255 for<br>each byte                          | a constant, 2 bytes                       |
| <b>K B 1)</b> | 0 to 255                                           | a constant, 1 byte                        |
| <b>K S 1)</b> | 2 alpha<br>characters                              | a constant, 2 ASCII characters            |
| <b>K G 1)</b> | + 0.1469368 x 10 - 38<br>to<br>+ 0.1701412 x 10+39 | a constant as floating point number       |
| <b>K T</b>    | 0.0 to 999.3                                       | a time (constant)                         |
| <b>K C 1)</b> | 0 to 999                                           | a count (constant)                        |

- 1) Not for transfers
- 2) PII Process input image
- 3) PIO Process output image
- 4) Only even parameters are allowed; error NNP is signalled for odd parameters.

The load and transfer operations are unconditional commands, i.e. they are executed irrespective of the result of the logic operation. The load and transfer operations can only be graphically programmed indirectly in conjunction with time or counting operations, otherwise only in statement lists.



**Example: Load and transfer function**



**Loading and transferring a time, also timing and counting operations**

| Given circuit | STEP 5 representation                                                                                                                        |                |                          |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                                                                                               | Ladder diagram | Control system flowchart |
|               | <pre>                     A I 5.0                     L IW 22                     SP T 10                     T QW 64                 </pre> |                |                          |

With graphic input, QW 64 was assigned to output DU of the timer. The programmer then automatically inserts the appropriate load and transfer command in the user program. Thus the contents of the memory location addressed with T 10 are loaded into the accumulator (Accu 1).

The accumulator contents (Accu 1) are then transferred to the process image addressed with QW 64.

### Loading and transferring input/output

The PLC user can load peripheral input bytes/words within the PLC program and transfer statuses to peripheral output bytes/words. (PY = byte, PW = word)

Beispiel: L PW 3  
T MW 199      Peripheral input word is loaded in ACCU 1 and the contents of this are transferred to flag word 199.

L MW 170  
T PW 18      The contents of MW 170 are loaded and transferred to peripheral output word 18.

The meaning of PY and PW depends therefore on whether they are written in conjunction with the load function or the transfer function

|                                  |                        |
|----------------------------------|------------------------|
| Load function (e.g.: L PW 3)     | inputs are addressed   |
| Transfer function (e.g. T PW 18) | outputs are addressed. |

### 9.3.4 Timing and counting operations

Timers T0-T16 and counters C0-C31 are enabled as standard. If a larger parameter is programmed the interpreter outputs the error message 6113 (Illegal MC5 type timer/counter). The number of counters can be increased to a maximum of 31 via PLC MD6. For this purpose MD6 must be used to specify the number of the last "active" MC5 timer (i.e. the one taken into account by the operating system). This number is effective from the next cold restart.

**Note:**

The way in which the operating system updates time cells does not permit programming of the 0.01s time grid.

In order to load a timer or counter with a set command, the value must be loaded in the accumulator beforehand.

The following load operations are expedient: <sup>1)</sup>

For timer: L KT, L IW, L QW, L FW, L DW  
 For counter: L KC, L IW, L QW, L FW, L DW

|                     | Operation | Parameter | Function                             |
|---------------------|-----------|-----------|--------------------------------------|
| Timing operations   | S I T     | 0 to 31   | Start a timer as a pulse             |
|                     | S V T     | 0 to 31   | Start a timer as an extended pulse   |
|                     | S E T     | 0 to 31   | Start a timer as an ON-delay         |
|                     | S S T     | 0 to 31   | Start a timer as a latching ON-delay |
|                     | S A T     | 0 to 31   | Start a timer as an OFF-delay        |
|                     | R T       | 0 to 31   | Reset a timer                        |
| Counting operations | S Z       | 0 to 31   | Set a counter                        |
|                     | R Z       | 0 to 31   | Reset a counter                      |
|                     | Z V Z     | 0 to 31   | Up counting                          |
|                     | Z R Z     | 0 to 31   | Down counting                        |

**Important note:**

Since the timer and counter commands are supported by the COP, and the latter has no parameter check, it is possible for the signal edge flags to be affected by timers or counters which are not programmed in this command.

**Example:**

As a result of a DO FW command, command SD T0 (substituted SD T 33) is to be processed, With RLO = 0: Bits ZWG, ZKS, FMS with T 1 are deleted.

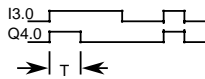
<sup>1)</sup> Timing or counting operations do not change the contents of accu 1.

**Pulse**

| Given circuit | STEP 5 representation                                   |                |                          |
|---------------|---------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                          | Ladder diagram | Control system flowchart |
|               | <pre> A  I 3.0 L  KT 10.2 SP T 1 A  T 1 =  Q 4.0 </pre> |                |                          |

With the first execution, the timer is started if the result of the logic operation is 1. If execution is repeated with  $RLO = 1$ , the timer is unchanged.  
If the  $RLO = 0$  the timer is set to zero (cleared).  
Scans AT and OT result in a logic 1 as long as the timer is still running.

DU and DE are digital outputs of the timer. The time is present with the timebase, binary-coded at output DU and BCD-coded at output DE.



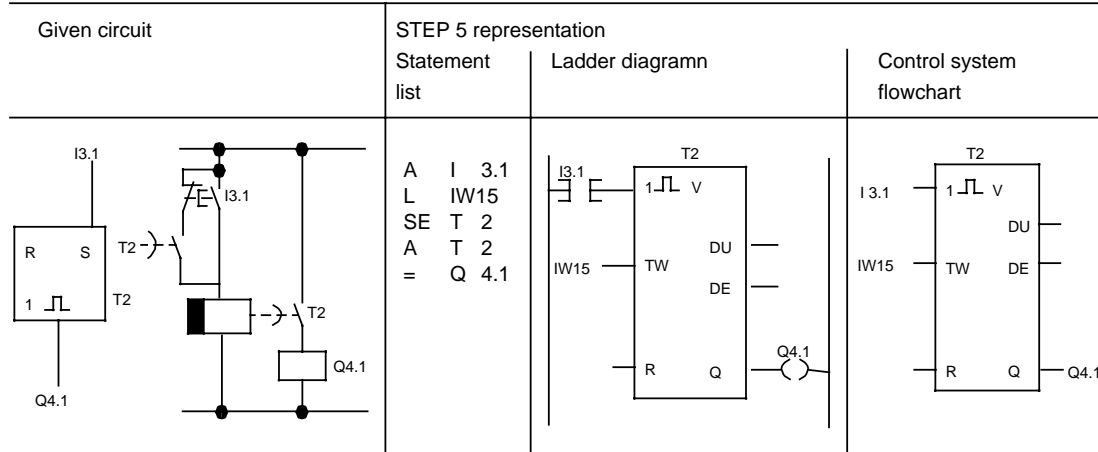
e.g.: KT10.2:

The specified value (10) is loaded in the timer. The number to the right of the point specifies the timebase:

- 1  $\hat{=}$  0.1 s
- 2  $\hat{=}$  1.0 s
- 3  $\hat{=}$  10.0 s

The programmed time equals  $10 \times 1.0\text{s} = 10\text{s}$ .

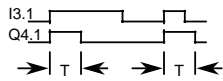
**Extended pulse**



With first execution, the timer is started if the result of the logic operation is 1.

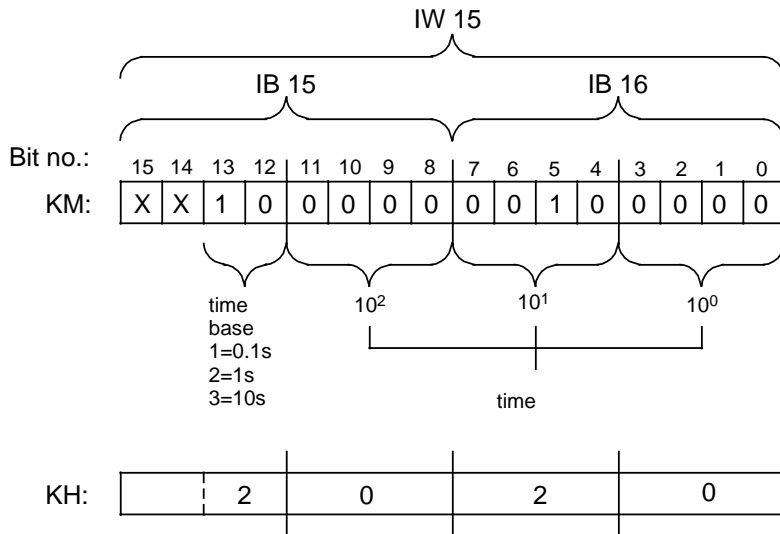
If the RLO = 0 the timer is unchanged.

Scans AT or OT result in a logic 1 as long as the timer is still running.



IW 15:

The timer is loaded with the value of operands I, Q, F or D present in BCD code (example IW15). The time reference is determined by bit nos. 12 and 13.



The programmed delay is 2x10x1.0s=20s

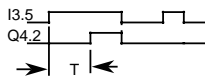
X = any value (0 or 1)

**ON-delay**

| Given circuit | STEP 5 representation                              |                |                          |
|---------------|----------------------------------------------------|----------------|--------------------------|
|               | Statement list                                     | Ladder diagram | Control system flowchart |
|               | <pre> A I 3.5 L KT 9.2 SD T 3 A T 3 = Q 4.2 </pre> |                |                          |

With the first execution, the timer is started if the result of the logic operation is 1. If execution is repeated and the RLO = 1, the timer is unchanged.  
If the RLO = 0 the timer is set to zero (cleared).

Scans AT or OT result in a logic 1 if the time has elapsed and the result of the logic operation is still present at the input.



KT 9.2:

The specified value (9) is loaded in the timer. The number to the right of the point specifies the timebase:

- 1   ≐   0.1s
- 2   ≐   1.0s
- 3   ≐   10.0s

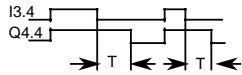
The programmed time equals  $9 \times 1.0\text{s} = 9\text{s}$ .

**OFF-delay**

| Given circuit | STEP 5 representation                                                 |                |                          |
|---------------|-----------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                        | Ladder diagram | Control system flowchart |
|               | <pre> AN I 3.4 L FW13 SF T 5 A T 5 = Q 4.4                     </pre> |                |                          |

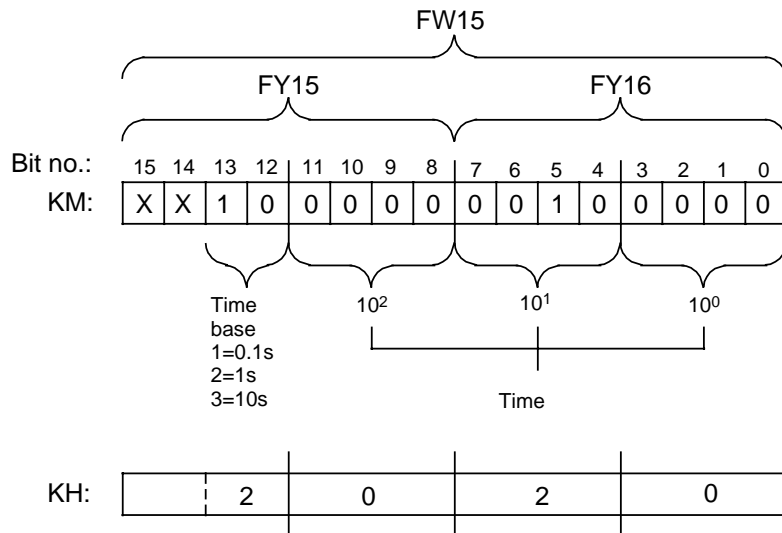
With the first execution, the timer is started if the result of the logic operation is 0. If execution is repeated and the RLO = 0, the timer is unchanged.  
 If the RLO = 1 the timer is set to zero (cleared).

Scans AT and OT result in a logic 1 if the time is still running or the RLO is still present at the input.



FW 13:

Setting of the time with the value of operands I, Q, F or D present in BCD code (flag word 13 in the example). The timebase is determined by bit 12 and bit 13.



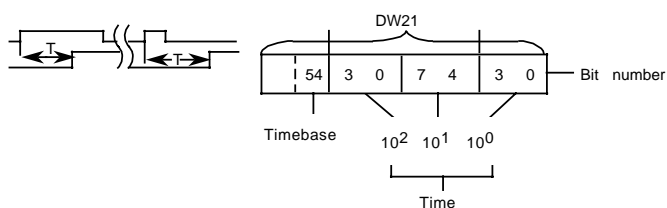
The programmed time equals  $2 \times 10 \times 1.0s = 20s$ .  
 X = optional (0 or 1)

### Latching ON-delay

| Given circuit | STEP 5 representation                                                              |                |                          |
|---------------|------------------------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                                     | Ladder diagram | Control system flowchart |
|               | <pre> A I 3.3 L DW21 SS T 4 A I 3.2 R T 4 A T 4 = Q 4.3                     </pre> |                |                          |

With the first execution, the timer is started if the result of the logic operation is 1.  
If the RLO = 0 the timer is unchanged.  
Scans AT and OT result in a logic 1 if the time has elapsed.

The logic level only goes to 0 when the timer has been reset with function RT.



#### DW 21:

Setting the time with the value of operands I, Q, F or D present in BCD code (data word 21 in the example).

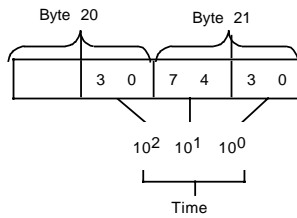


### Setting a counter

| Given circuit | STEP 5 representation                                                                                         |                |                          |
|---------------|---------------------------------------------------------------------------------------------------------------|----------------|--------------------------|
|               | Statement list                                                                                                | Ladder diagram | Control system flowchart |
|               | <pre>                     A I 4.1                     L IW20                     S C 1                 </pre> |                |                          |

With the first execution, the counter is set if the result of the logic operation is 1. If execution is repeated, the counter is unchanged (irrespectively of whether the RLO is 1 or 0). With repeated first execution with RLO = 1, the counter is set again (signal edge decoding). DU and DE are digital outputs of the counter. The count is present in binary code at output DU, and BCD-coded at output DE.

The flag required for signal edge decoding of the set input is also present in the count word.



IW 20:

Setting a counter with the value of operands I, Q, F or D present in BCD code (input word 20 in the example).

**Resetting a counter**

| Given circuit | STEP 5 representation                    |                |                          |
|---------------|------------------------------------------|----------------|--------------------------|
|               | Statement list                           | Ladder diagram | Control system flowchart |
|               | <pre> A I 4.2 R C 1 A C 1 = Q 2.4 </pre> |                |                          |

If the result of the logic operation is 1 the counter is set to zero (cleared).  
If the result of the logic operation is 0 the counter is unchanged.

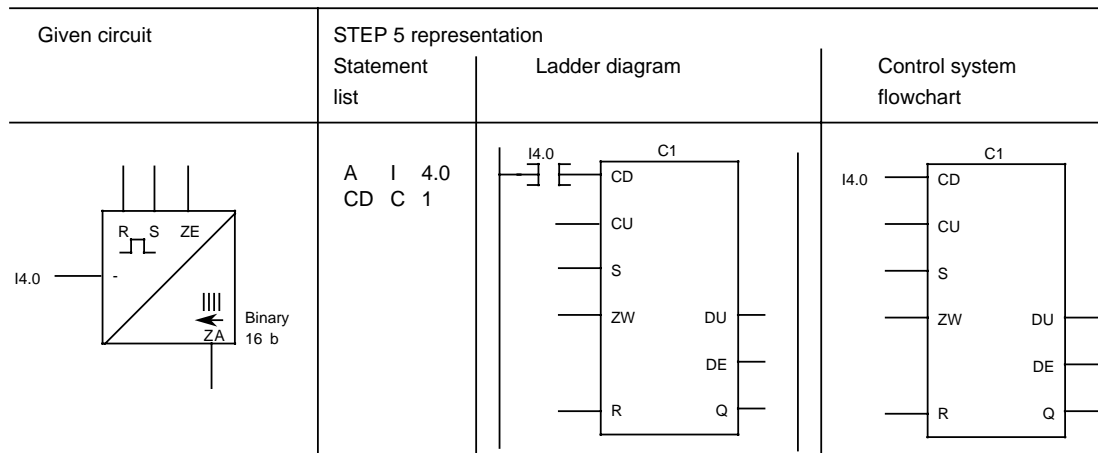
**Up counting**

| Given circuit | STEP 5 representation       |                |                          |
|---------------|-----------------------------|----------------|--------------------------|
|               | Statement list              | Ladder diagram | Control system flowchart |
|               | <pre> A I 4.1 CU C 1 </pre> |                |                          |

The value of the addressed counter is incremented by 1. Function CU (count up) is only executed with a positive-going edge (from 0 to 1) of the logic operation programmed before CU. The flags required for signal edge decoding of the count inputs are also contained in the count word.

A counter with two different inputs can be used as an up/down counter by means of the two separate signal edge flags for CU (count up) and CD (count down).

**Down counting**



The value of the addressed counter is decremented by 1. The function only becomes effective with a positive-going edge (from 0 to 1) of the logic operation programmed before CD. The flags required for signal edge decoding of the count inputs are also in the count word. A counter with two different inputs can be used as an up/down counter by means of the two separate signal edge flags for CU (count up) and CD (count down).

**9.3.5 Comparison operations**

The comparison operations compare the content of Accumulator 1 with the content of Accumulator 2. The values to be compared must therefore first be stored in the accumulators, e.g. with load operations. The accumulator contents remain unchanged during the comparison.

| Operation                    | Parameter | Function                                                       |
|------------------------------|-----------|----------------------------------------------------------------|
| ! = <input type="checkbox"/> |           | Test if equal                                                  |
| >< <input type="checkbox"/>  |           | Test if not equal                                              |
| > <input type="checkbox"/>   |           | Test if greater                                                |
| > = <input type="checkbox"/> |           | Test if greater than or equal to                               |
| < <input type="checkbox"/>   |           | Test if less                                                   |
| < = <input type="checkbox"/> |           | Test if less than or equal to                                  |
| ↑<br>F<br>D                  |           | Two fixed-point numbers<br>Two fixed-point double-word numbers |

**Test if equal**

| Given circuit | STEP 5 representation                 |                |                          |
|---------------|---------------------------------------|----------------|--------------------------|
|               | Statement list                        | Ladder diagram | Control system flowchart |
|               | <pre>L IB19 L IB20 != F = Q 3.0</pre> |                |                          |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

RLO = 1: Comparison is fulfilled, Accu 1-L = Accu 2-L

RLO = 0: Comparison is not fulfilled, Accu 1-L = Accu 2-L

Accu 2-H and Accu 1-H are not involved in the operation with the fixed point comparison. The numeric representation of the operands (fixed-point calculation) is taken into account in the comparison.

|   |      |          |
|---|------|----------|
| 0 | IB19 | Accu 2-L |
| 0 | IB19 | Accu 1-L |

**Test if not equal**

| Given circuit | STEP 5 representation                      |                |                          |
|---------------|--------------------------------------------|----------------|--------------------------|
|               | Statement list                             | Ladder diagram | Control system flowchart |
|               | <pre>L IB21 L DW3 &gt;&gt; F = Q 3.1</pre> |                |                          |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

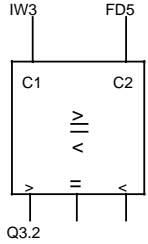
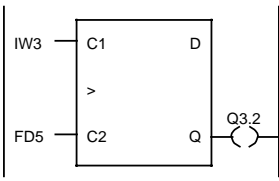
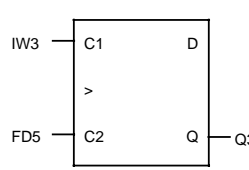
RLO = 1: Comparison is fulfilled, Accu 1-L = Accu 2-L

RLO = 0: Comparison is not fulfilled, Accu 1-L = Accu 2-L

Accu 2-H and Accu 1-H are not involved in the operation with the fixed point comparison. The numeric representation of the operands (fixed-point calculation in this case) are taken into account in the comparison.

|   |      |          |
|---|------|----------|
| 0 | IB21 | Accu 2-L |
|   | DW3  | Accu 1-L |

**Test if greater**

| Given circuit                                                                     | STEP 5 representation                 |                                                                                   |                                                                                     |
|-----------------------------------------------------------------------------------|---------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|                                                                                   | Statement list                        | Ladder diagram                                                                    | Control system flowchart                                                            |
|  | <pre>L IW3 L FD5 &gt; D = Q 3.2</pre> |  |  |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

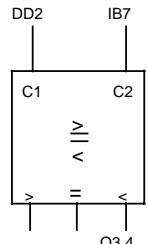
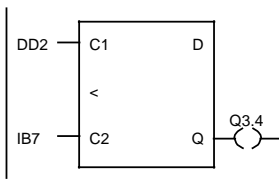
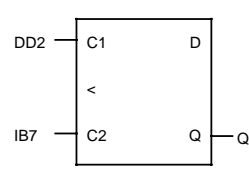
RLO = 1: Comparison is fulfilled, Accu 2 > Accu 1

RLO = 0: Comparison is not fulfilled, Accu 2 < Accu 1

|          |     |     |     |     |          |
|----------|-----|-----|-----|-----|----------|
| Accu 2-H | 0   | 0   | IB3 | IB4 | Accu 2-L |
| Accu 1-H | FB5 | FB6 | FB7 | FB8 | Accu 1-L |

The numeric representation of the operands is taken into account in the comparison, i.e. the contents of Accu 1 and Accu 2 are interpreted as fixed-point numbers with double-word width.

**Test if less**

| Given circuit                                                                       | STEP 5 representation                 |                                                                                     |                                                                                       |
|-------------------------------------------------------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
|                                                                                     | Statement list                        | Ladder diagram                                                                      | Control system flowchart                                                              |
|  | <pre>L DD2 L IB7 &lt; D = Q 3.4</pre> |  |  |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

RLO = 1: Comparison is fulfilled, Accu 2 < Accu 1

RLO = 0: Comparison is not fulfilled, Accu 2 > Accu 1

|          |     |     |          |
|----------|-----|-----|----------|
| Accu 2-H | DW2 | DW3 | Accu 2-L |
| Accu 1-H | 0   | 0   | IB7      |

The numeric representation of the operands is taken into account in the comparison, i.e. the contents of Accu 1 and Accu 2 are interpreted as fixed-point numbers with double-word width.

**Test if greater than or equal to**

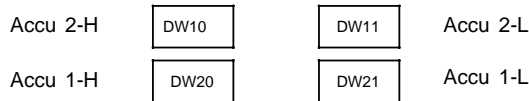
| Given circuit                                                                                                                                                                                                                                                                                                | STEP 5 representation             |                                                                                                                                                                          |                                                                                                                                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                                                                                              | Statement list                    | Ladder diagram                                                                                                                                                           | Control system flowchart                                                                                                                               |
| <p>The given circuit shows two comparators, C1 and C2, connected in series. C1 compares the value in register DD10 (input from the left) with the value in register DD20 (input from the right). The output of C1 is connected to the input of C2. The output of C2 is connected to a coil labeled Q3.3.</p> | <pre>L DD10 L DD20 &gt;=G =</pre> | <p>The ladder diagram shows two normally closed contacts, C1 and C2, connected in series. C1 is labeled with DD10 and C2 with DD20. The output coil is labeled Q3.3.</p> | <p>The control system flowchart shows two input boxes labeled C1 and C2. C1 is labeled with DD10 and C2 with DD20. The output box is labeled Q3.3.</p> |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

RLO = 1: Comparison is fulfilled, Accu 2  $\geq$  Accu 1

RLO = 0: Comparison is not fulfilled, Accu 2 < Accu 1



The numeric representation of the operands is taken into account in the comparison, i.e. the contents of Accu 1 and Accu 2 are interpreted as a floating-point number.

**Test if less than or equal to**

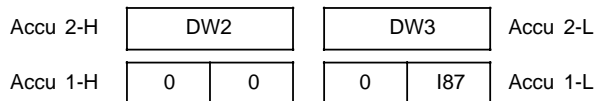
| Given circuit                                                                                                                                                                                                                                                                                              | STEP 5 representation                 |                                                                                                                                                                        |                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                                                                                            | Statement list                        | Ladder diagram                                                                                                                                                         | Control system flowchart                                                                                                                             |
| <p>The given circuit shows two comparators, C1 and C2, connected in series. C1 compares the value in register DD2 (input from the left) with the value in register IB7 (input from the right). The output of C1 is connected to the input of C2. The output of C2 is connected to a coil labeled Q3.4.</p> | <pre>L DD2 L IB7 &lt; G = Q 3.4</pre> | <p>The ladder diagram shows two normally closed contacts, C1 and C2, connected in series. C1 is labeled with DD2 and C2 with IB7. The output coil is labeled Q3.4.</p> | <p>The control system flowchart shows two input boxes labeled C1 and C2. C1 is labeled with DD2 and C2 with IB7. The output box is labeled Q3.4.</p> |

The operand first specified is compared with the next operand according to the comparison function.

The comparison results in a binary result of the logic operation.

RLO = 1: Comparison is fulfilled, Accu 2  $\leq$  Accu 1

RLO = 0: Comparison is not fulfilled, Accu 2 > Accu 1



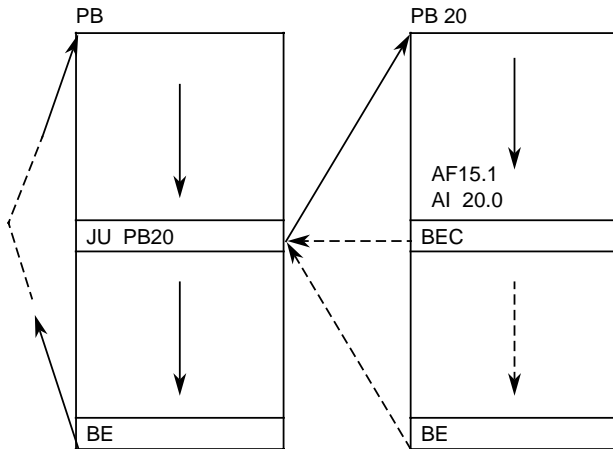
The numeric representation of the operands is taken into account in the comparison, i.e. the contents of Accu 1 and Accu 2 are interpreted as a floating-point number.

### 9.3.6 Block calls

| Operation                                                                                            | Parameter | Function                                                                                                                                           |
|------------------------------------------------------------------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>JU</b> □□<br><b>JC</b> □□<br>↑↑<br><b>PB</b> 1 to 255<br><b>FB</b> 1 to 255<br><b>SB</b> 1 to 255 |           | Unconditional jump<br>Conditional jump<br>(depending on the RLO)<br><br>to a program block<br>to a function block (type FB)<br>to a sequence block |
| <b>C DB</b>                                                                                          | 1 to 255  | Data block call                                                                                                                                    |
| <b>BE</b><br><b>BEC</b><br><b>BEU</b>                                                                |           | Block end<br>Block end, conditional (depending on RLO)<br>Block end, unconditional                                                                 |

Command C DB (data block call) is explained under "Calling data blocks" (Page 3-1).  
 Commands BE (block end) and BEC (block end, conditional) result in a return to the previously nested block.  
 Command BE must be programmed at the end of each data block (except for DB).

**Example:**



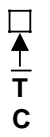
If the result of the logic operation is 1, the return to PB7 already takes place with processing of the BEC command.  
 If the result of the logic operation is 0, processing of PB20 continues up to the BE command, which then initiates the return to PB7 when PB20 has been fully processed.



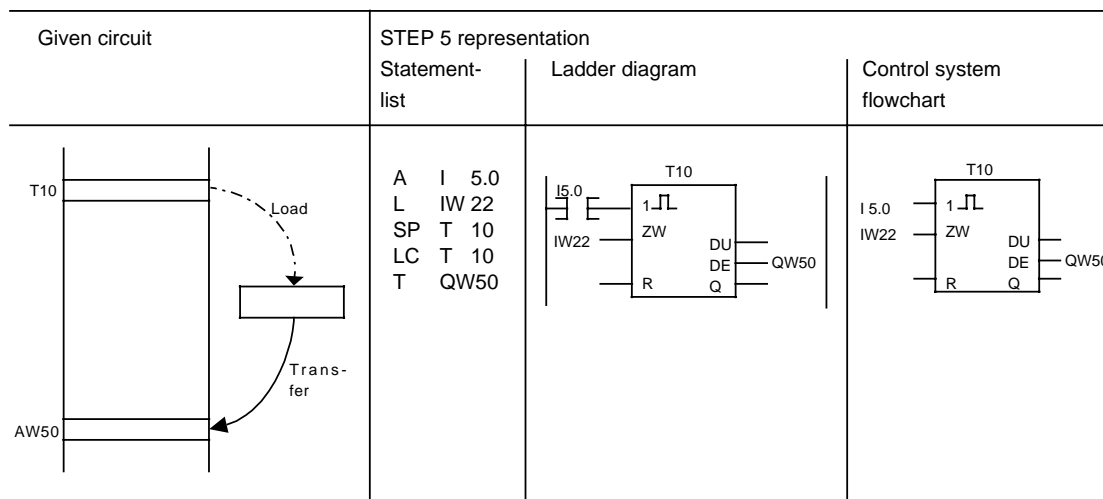


### 9.3.7 Code operations

The code operations allow a time or count which is present in binary form, to be loaded as a code in the accumulator; the corresponding value is still available in BCD form for further processing.

| Operation                                                                                   | Parameter           | Function                            |
|---------------------------------------------------------------------------------------------|---------------------|-------------------------------------|
| <b>LC</b>  | 1 to 127<br>1 to 63 | Load as code<br><br>times<br>counts |

#### Loading a time (coded)



The content of the memory location addressed with T10 is loaded as a code in the accumulator.

The subsequent transfer operation transfers the content from the accumulator to the memory location of the process images addressed with QW50. With the graphic methods of representation LAD and CSF a coding operation can only take place indirectly as a result of the assignment of output DE of a timer or counter. With method of representation STL, however, this command can be isolated.

### 9.3.8 Arithmetic operations

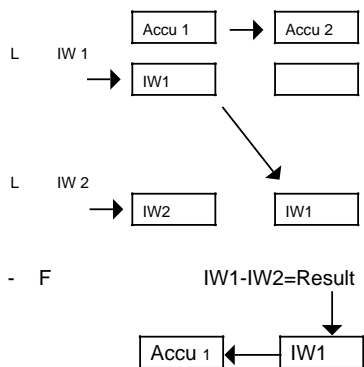
Arithmetic operation can only be represented in the statement list.

They process the contents of Accumulators 1 and 2. Suitable load operations, for example, are required.

| Operation  | Parameter | Function                                                  |
|------------|-----------|-----------------------------------------------------------|
| + F<br>- F |           | Addition<br>Subtraction<br><br>of two fixed-point numbers |

By means of two load operations, Accumulators 1 and 2 can be loaded according to the operands of the load operations. Arithmetic operations can then be executed with the contents of both accumulators.

#### Example:



The subsequent transfer operation transfers the result stored in Accu 1 to the operand issued for the transfer operation.

If, during calculation with fixed-point numbers, an overflow occurs ( $OV = 1$ ) Accu 1-H is cleared.

### 9.3.9 Other operations

The following operations can only be represented in the statement list

| Operation      | Parameter | Function                        |
|----------------|-----------|---------------------------------|
| <b>STP</b>     |           | Stop                            |
| <b>NOP 0</b>   |           | No operation (all bits cleared) |
| <b>NOP 1</b>   |           | No operation (all bits set)     |
| <b>BLD 255</b> |           | Screen command                  |

The STOP command is used, for example, when the PLC is required to go to the stop state in the event of certain critical states of the system or when a device error occurs.

The no-operations serve, for example, for keeping memory locations free or overwriting them.

The screen command governs the subdivision of program parts into segments (networks) within a block. It is automatically stored in the program by the programmer and is treated as a no-operation by the interface controller.

The programmer can also insert "BLD 255" into an already written block as an end of segment.

### 9.4 Supplementary operation (with function blocks only)

Function blocks can be programmed with an operation set which is extended compared to the program blocks. The full operation set for function blocks comprises the basic operations and the supplementary operations.

With the function blocks, the operations are only represented in a statement list. The programs of the function blocks therefore cannot be programmed in graphic form (CSF or LAD).

Described in the following are the supplementary operations. Possibilities of combination of the substitution commands with the actual operands are also given.

### 9.4.1 Logic operations, binary

| Operation                       | Description                                                                                                                                                                                               |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>A =</b> <input type="text"/> | <b>AND function</b> to test a formal operand for logic 1                                                                                                                                                  |
| <b>AN=</b> <input type="text"/> | <b>AND function</b> to test a formal operand for logic 0                                                                                                                                                  |
| <b>O =</b> <input type="text"/> | <b>OR function</b> to test a formal operand for logic 1                                                                                                                                                   |
| <b>ON=</b> <input type="text"/> | <b>OR function</b> to test a formal operand for logic 0                                                                                                                                                   |
|                                 | <p><b>Insert formal operand</b><br/>The actual operands allowed are binary addressed inputs, outputs and flags (parameters: I, O; parameter type DI) as well as timers and counters (parameters T,C.)</p> |

#### Example:

```

: A -ON
: AN -STOP
: AN -END
: O -AMNT
:= -RUN

```

### 9.4.2 Setting operations

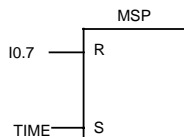
| Operation                       | Description                                                                                                                                              |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>S =</b> <input type="text"/> | Set (binary) a formal operand                                                                                                                            |
| <b>RB=</b> <input type="text"/> | Reset (binary) a formal operand                                                                                                                          |
| <b>==</b> <input type="text"/>  | Assign the result of the logic operation to a formal operand                                                                                             |
|                                 | <p><b>Insert formal operand</b><br/>The actual operands allowed are binary addressed inputs, outputs and flags (parameters: I, Q; parameter type DI)</p> |

#### Example:

```


Q I 0.7
RB = MSP
Q = TIME
S = MSP

```



The diagram shows a rectangular block representing a timer. It has two input lines on the left: the top one is labeled 'R' and is connected to 'I0.7'; the bottom one is labeled 'S' and is connected to 'TIME'. On the right side of the block, there is an output line labeled 'MSP'.

### 9.4.3 Timing and counting operations

| Operation                                                                           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RD=</b> <input type="text"/>                                                     | Reset (digital) a formal operand<br>(parameters: T,C)                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>SP=</b> <input type="text"/>                                                     | Start a time, preset as formal operand, with the value stored in the accumulator as a pulse<br>(parameter: T)                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SR=</b> <input type="text"/>                                                     | Start a time, preset as formal operand, with the value stored in the accumulator as an on-delay<br>(parameter: T)                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>SEC=</b> <input type="text"/>                                                    | Start a time, preset as formal operand, with the value stored in the accumulator as an extended pulse; or set a counter, preset as formal operand, with the following, specified count<br>( parameters: T, C)                                                                                                                                                                                                                                                                                                                     |
| <b>SSV=</b> <input type="text"/>                                                    | Start a time, preset as formal operand, with the value stored in the accumulator as a latching on delay, or up-counting of a counter specified as formal operand<br>(parameters: T, C)                                                                                                                                                                                                                                                                                                                                            |
| <b>SFD=</b> <input type="text"/>                                                    | Start a time, preset as formal operand, with the value stored in the accumulator as an off-delay, or down-counting of a counter preset as formal operand<br>(parameters: T, C)                                                                                                                                                                                                                                                                                                                                                    |
|  | <p><b>Insert formal operand</b> (see Page 8)<br/>                     The actual operands allowed are timers and counters; exception: Timers only with SP and SR.<br/>                     The time or count can be specified as follows, as for the basic operations or as a formal operand:<br/>                     Set the time or count with the value present in BCD code of operands IW, QW, FW, DW (parameters: I; parameter type: W) specified as formal operands, or as data (parameter: D; parameter type: KT, KC)</p> |

**Examples:**

| Function block call                                                                                     | Program in function block                                                                                           | Program executed                                                                                         |
|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <pre> :JU FB203 NAME :EXAMPLE ANNA : I 10.3 BERT : T 17 HANS : Q 18.4 </pre>                            | <pre> : A = ANNA : L KT 010.2 : SSU = BERT : A = BERT := = HANS </pre>                                              | <pre> : A I 10.3 : L KT 010.2 : A T 17 : A T 17 := Q 18.4 </pre>                                         |
| <pre> :JU FB204 NAME :EXAMPLE MAXI : I 10.5 IRMA : I 10.6 EVA : I 10.7 DORA : C 15 EMMA : F 58.3 </pre> | <pre> : A = MAXI : SSU = DORA : A = IRMA : SFD = DORA : A = EVA : L KZ100 : SEC = DORA : AN = DORA := = EMMA </pre> | <pre> : A I 10.5 : CU C 15 : A I 10.6 : CD C 15 : A I 10.7 : L KC100 : S C 15 : AN C 15 := F 58.3 </pre> |
| <pre> :JU FB205 NAME :EXAMPLE KURT : I 10.4 CARL : T 18 EGON : IW20 MAUS : F 100.7 </pre>               | <pre> : A = KURT : L = EGON : SEC = CARL := = MAUS </pre>                                                           | <pre> : A I 10.4 : L IW20 : SE T 18 := F 100.7 </pre>                                                    |

### 9.4.4 Enabling operations for timing and counting operations

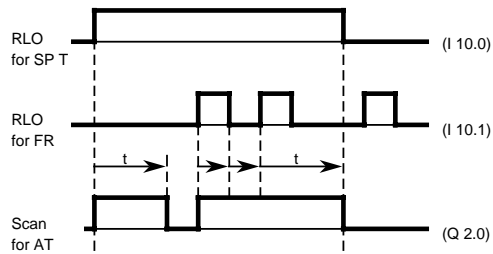
| Operation                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>FR T 0 to 127</b><br><br><b>FR</b> | <b>Enabling a time for a restart</b><br>The operation is only executed with a leading edge of the result of the logic operation. It initiates a restart of the time if the RLO present is 1 for the start operation.<br><br><b>C0 to 127 Enabling a counter</b><br>The operation is only executed with a leading edge of the result of the logic operation. It initiates setting, up or down counting of the counter if the RLO present is 1 for the corresponding operation. |
| <b>FR=</b> <input type="text"/>       | <b>Enabling a formal operand for a restart,</b><br>parameters (T, C)                                                                                                                                                                                                                                                                                                                                                                                                          |

**Example:**

```

: A I 10.0
: L KT 500.0
: SP T 10
: A I 10.1
: FR T 10
: A T 10
:= Q 2.0

```



### 9.4.5 Bit test operations

| Operation                           | Parameter     | Function                  |
|-------------------------------------|---------------|---------------------------|
| <b>TB</b> <input type="checkbox"/>  |               | Test the bit for logic 1  |
| <b>TBN</b> <input type="checkbox"/> |               | Test the bit for logic 2  |
| <b>SU</b> <input type="checkbox"/>  |               | Set bit unconditionally   |
| <b>RU</b> <input type="checkbox"/>  |               | Reset bit unconditionally |
| <b>I</b>                            | 0.0 to 127.7  | an input                  |
| <b>Q</b>                            | 0.0 to 127.7  | an output                 |
| <b>F</b>                            | 0.0 to 255.7  | a flag                    |
| <b>C</b>                            | 0.0 to 127.15 | a count word              |
| <b>T</b>                            | 0.0 to 127.15 | a time word               |
| <b>D</b>                            | 0.0 to 255.15 | a data word               |

Operations "P" and "PN" are scans. They test a bit of the operand specified in the following, and then insert the result of the logic operation irrespective of previous scans and the previous status.

| Operation  | Logic level of the bit in the specified operand | Result of logic operation |
|------------|-------------------------------------------------|---------------------------|
| <b>TB</b>  | 0                                               | 0                         |
|            | 1                                               | 1                         |
| <b>TBN</b> | 0                                               | 1                         |
|            | 1                                               | 0                         |

The RLO formed in this way can be subjected to further logic operations. However, a bit test operation must always be positioned at the beginning of a logic operation.

#### 1st example

The logic level of bit no. 10 of data word 205 is ANDed with the logic level of input I13.7.

```
: C DB 200
: TB D 205.10
: A I 13.7
:= F 210.3
```

Operations "SU" and "RU" are executed independently of the result of the logic operation. When this operation has been processed, the addressed bit in the specified operand is set to logic 1 (for SU) or logic 0 (for RU).


#### 2nd example

The bit no. 3 is to be set by DW55 and bit no. 11 by DW103.

```
: SU D 55.3
: RU D 103.11
```



### 9.4.6 Coding, load and transfer operations

| Operation                                                                                                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>L =</b> <input type="text"/>                                                                                       | <b>Load a formal operand</b><br>The value of the operand specified as a formal operand will be loaded into the accu (parameters: I, Q; parameter type: BY W, D).                                                                                                                                                                                                                                                                                                                                                  |
| <b>LC=</b> <input type="text"/>                                                                                       | <b>Load a formal operand as code</b><br>The value of the timer or counter specified as a formal operand will be loaded into the accu in BCD form (parameters: T, C).                                                                                                                                                                                                                                                                                                                                              |
| <b>LW=</b> <input type="text"/>                                                                                       | <b>Load the bit pattern of a formal operand</b><br>The bit pattern of the formal operand will be loaded into the accu (parameter : D; parameter type: KF, KH, KM, KY, KS, KT, KC).                                                                                                                                                                                                                                                                                                                                |
| <b>LDW=</b> <input type="text"/>                                                                                      | <b>Load the bit pattern of a formal operand</b><br>The bit pattern of the formal operand will be loaded into the accu (parameter: D; parameter type: KG).                                                                                                                                                                                                                                                                                                                                                         |
| <b>T =</b> <input type="text"/><br> | <b>Transfer to a formal operand</b><br>The accumulator content will be transferred to the operand specified as formal operand (parameters: I,Q; parameter type: BY, W, D).<br><br><b>Insert formal operand</b><br>The operands corresponding to the basic operations are allowed as actual operands. The data allowed for LW is in the form of a binary pattern, hex pattern, two-byte absolute numbers, characters, fixed-point number, times and counts.<br>For LD, a floating-point number is allowed as data. |

**Example:**

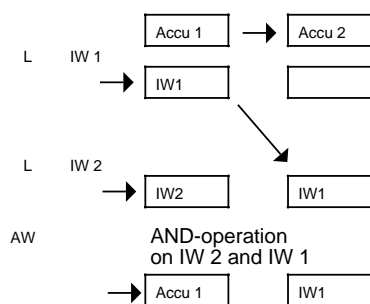
| Function block call                                        | Program in function block    | Program execution               |
|------------------------------------------------------------|------------------------------|---------------------------------|
| : JU FB206<br><br>NAME : COMP.<br>C1 : KH7F0A<br>C2 : DW20 | : LW =C1<br>: L =C2<br>: !=F | : L KH7F0A<br>: L DW20<br>: !=F |

### 9.4.7 Logic operations, digital

| Operation  | Description                                        |
|------------|----------------------------------------------------|
| <b>AW</b>  | AND operation, digital, Accu 1 and Accu 2          |
| <b>OW</b>  | OR operation, digital, Accu 1 and Accu 2           |
| <b>XOW</b> | Exklusive OR operation, digital, Accu 1 and Accu 2 |

Accu 1 and Accu 2 can be loaded according to the operands of the load operation, by means of two load operations (see also Page 9-41). The contents of both accumulators can then be subjected to a digital operation.

#### Example:



The subsequent transfer operation transfers the result stored in Accu 1 to the operand specified with the transfer operation.

### 9.4.8 Shift operations

| Operation          | Description                                                               |
|--------------------|---------------------------------------------------------------------------|
| <b>SLW 0 to 15</b> | Shift left (zeros are shifted in from the right)                          |
| <b>SRW 0 to 15</b> | Shift right (zeros are shifted in from the left)                          |
| <b>SSW 0 to 15</b> | Shift right with sign (the sign is shifted in from the left)              |
| <b>SLD 0 to 32</b> | Shift left, double word (zeros are shifted in from the right)             |
| <b>SSD 0 to 32</b> | Shift right with sign, double word (the sign is shifted in from the left) |

The shift functions are executed independently of conditions. The last bit to be shifted can be scanned with jump functions. JZ can be used for the jump if the bit is 0, and JN or JC if the bit is 1.

#### Example:

STEP 5 program: Contents of data words

```
:L DW52 H = 14AF
:SLW 4
:T DW53 H = 4AF0
```

### 9.4.9 Conversion operations

| Operation                | Meaning                                                      |
|--------------------------|--------------------------------------------------------------|
| <b>CFW</b><br><b>CSW</b> | One's complement of Accu 1-L<br>Two's complement of Accu 1-L |

**Examples:**

The contents of data word 64 are to be inverted bit for bit and stored in data word 78.

STEP 5 program: Assignments of data words:

```
: L DW64 KM = 0011 1110 0101 1011
: CFW
: T DW78 KM = 1100 0001 1010 0100
```

The contents of data word 207 are to be interpreted as a fixed-point number and stored in data word 51 with the opposite sign.

STEP 5 program: Assignments of data words:

```
: L DW207 KF: +51
: CSW
: T DW51 KF: - 51
```

### 9.4.10 Decrementing/incrementing

| Operation                              | Description                                                                                                                                                                                                                                      |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>D 1 to 255</b><br><b>I 1 to 255</b> | Decrementing<br>Incrementing<br>Accumulator contents 1 are decremented/incremented by the number specified in the parameter. Execution of the operation is independent of conditions. It is restricted to the <b>right byte</b> (without carry). |

**Example:**

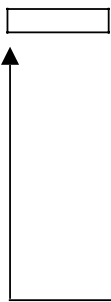
STEP 5 program: Assignments of data words:

```
: LDW7 KH = 1010
: I 16
: T DW8 KH = 1020
: D 33
: T DW9 KH = 10FF
```

## 9.4.11 Jump operations

The jump destination for unconditional and conditional jumps is specified symbolically (maximum of 4 characters): The symbolic parameter of the jump command is identical with the symbolic address of the statement to be jumped to. When programming, ensure that the unconditional jump distance is not more than  $\pm 127$  words. It should be noted that a STEP 5 statement must not comprise more than one word. Jumps may only be executed within a block. Jumps extending beyond networks are not allowed.

| Operation                        | Description                                                                                                                                                                                                                                    |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>JU</b> = <input type="text"/> | <b>Jump, unconditional</b><br>The unconditional jump will be executed independently of conditions.                                                                                                                                             |
| <b>JC</b> = <input type="text"/> | <b>Jump, conditional</b><br>The conditional jump will be executed if the result of the logic operation is 1. If the RLO is 0 the statement will not be executed and the RLO will be set to 1.                                                  |
| <b>JZ</b> = <input type="text"/> | <b>Jump if accumulator content is zero</b><br>The jump will be executed if the accumulator content is zero. If the accumulator content is not zero the jump will not be executed. The RLO will not be changed.                                 |
| <b>JN</b> = <input type="text"/> | <b>Jump if accumulator content is not zero</b><br>The jump will be executed if the accumulator content is not zero. If the accumulator content is zero the jump will not be executed. The RLO will not be changed.                             |
| <b>JP</b> = <input type="text"/> | <b>Jump if accumulator content is positive</b><br>The jump will be executed if the accumulator content is greater than zero. If the accumulator content is zero or less than zero, the jump will not be executed. The RLO will not be changed. |
| <b>JM</b> = <input type="text"/> | <b>Jump if accumulator content is negative</b><br>The jump will be executed if the accumulator content is less than zero. If the accumulator content is zero or greater than zero, the jump will not be executed. The RLO will not be changed. |
| <b>JO</b> = <input type="text"/> | <b>Jump if overflow</b><br>The jump will be executed in the event of an overflow. If there is no overflow, the jump will not be executed. The RLO will not be changed.                                                                         |

| Operation                                                                                                                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>JOS =</b> <input type="text"/></p>  | <p><b>The jump will be executed if "Overflow stored" is set (OS = 1)</b><br/>                     Otherwise (OS = 0) the jump will not be executed.<br/>                     "Overflow stored" will be set for arithmetic operations in the event of an overflow, and will remain stored until the arithmetic operation is interrupted. An overflow exists when, with numeric representation, the permissible range is exceeded by an arithmetic operation.</p> <p><b>Insert symbolic address</b> (maximum of 4 characters).</p> |

The conditional jump operations (all except for JU) are executed in accordance with the RLO and the indicators in the control unit of the PLC.

**Note:**

The jump statement and jump destination must be located in a network. Only one symbolic address is allowed for jump destinations per network.

**Example:**

```

 : JU = FORT
 :
FORT : A - STOP
 : A - END
 :
ZIEL : O Q 7.3
 : O F 16.6
 :
 : O - BETR
 : JC = ZIEL

```

**Example (comparison operations):**

```

 : L DW 67
 : L DW 107
 : !=F
 : JC = GLCH (Jump if equal, JZ can also be programmed)
 : JM = KLNR (Jump if less)
 : JP = GRSS (Jump if greater)
 :
KLNR :
 :
GLCH :
 :
GRSS :

```

**Example (arithmetic operations):**

```

: L WERT
: L - MESS
:+F
: JZ = ZERO (Jump if zero)
: JP = POSI (Jump if positive)
:
:
POSI :
:
ZERO :

```

**Example (digital operation):**

```

: L FW25
: L IW10
: XOW
: JZ = ZERO (Jump if accu content = KH 0000)
: JN = NONZ (Jump if accu content = KH 0000)
:
:
ZERO :
:
NONZ :

```

**Example ( shift operations):**

```

: L QW101
: SLW 10
: JZ = NULL (Jump if last shifted bit = 0)
: JN = EINS (Jump if last shifted bit = 1)
:
:
NULL :
:
EINS :
:

```

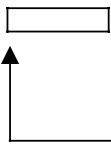
**Example (conversion operations):**

```

: L PW169
: KZW
: JO = OVFL (Jump if overflow)
: JN = NONZ (Jump if accu content = KH 0000)
:
:
NONZ :
:
OVFL :
:

```

### 9.4.12 Processing operations

| Operation                                                                                                            | Description                                                                                                                                                                                                                                                               |
|----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DO=</b> <input type="text"/><br> | <b>Process formal operand</b> (type of parameter: DO)<br><br><b>Insert formal operand</b> (see page 4-5)<br><br>Only the following operations can be substituted:<br>C DB<br>JU PB<br>JU SB<br>JU FB<br>(See "Type of block parameter and allowed actual operand, page 7) |
| <b>DO DW0 to 254</b><br>(Operation)                                                                                  | <b>Process data word*</b><br>The specified operation which follows will be combined with the parameter given in the data word and executed.                                                                                                                               |
| <b>DO FW0 to 254</b>                                                                                                 | <b>Process flag word*</b><br>The specified operation which follows will be combined with the parameter given in the flag word and executed.                                                                                                                               |

\*Two-word commands can also be substituted. The following command sequences, for example, are therefore possible:

```

C DB20 C DB100
.
.
.
.
.
.
DO FW240 DO DW21
TB D 0.0 S C 0.0

```

or

The address of the bit actually addressed must be stored, as usual, in the corresponding pointer word (FW240 or DW21 in the example). The byte/word address must be stored on the right and the bit address on the left. Any bits which are beyond the bit address in the high byte of the pointer will be deleted.

When substituting two-word commands in the process image, the following should be noted:

- The distinction between inputs and outputs is not made in the opcode but in the address part of the command i.e. the specification must be made in the pointer word.

**Example:**

```

L KH 0104
T FW 250
.
.
DO FW 250
SU I 0.0 (or SU Q 0.0)
->I 4.1 will be set

```

- If Q 4.1 is to be set, the value KH0184 must be stored in the flag word.

- In the specification of command F, the two expressions SU I 0.0 or SU Q 0.0 are fully equivalent.
- Any self-programmed bit address in the command (e.g. SU I 4.7) will be ignored.

### Example: Process data word

The contents of data words DW 20 to DW 100 are to be deleted. The "index register" for the parameter of the data words is DW 0.

```

 : L KF 20 Assignment for "index register"
 : T DW1
M1 : L KF 0 Reset
 : DO DW 1
 : T DW 0
 : L DW 1 Increment the index register
 : L KF 1
 :+F
 : T DW1
 : L KF 100
 : <=F
 : JC =M1 Jump if the index is within the range
 Other STEP 5 program

```

The following operations can be combined with DO DW or DO FW:

|                                                                                                                                                                                |                                                                                                                                                                                                   |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| A, AN, O, ON<br>S, R, = .<br>FRT, R T, SF T, SD T, SI T, SS T, SE T<br>FRC, R C, S C, CD C, CU C<br>L, LD, T<br>JU, JC, JZ, JN, JP, JM, JO<br>SLW, SRW<br>D, I<br>C DB, JU, JC | Binary operations<br>Storage functions<br>Time functions<br>Counting functions<br>Loading and transfer function<br>Jump functions<br>Shift functions<br>Decrementing, incrementing<br>Block calls |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

The programmer does not verify the validity of the combination. No two or three-word commands and no operations with formal operands may be combined in function blocks.

### Example: Process data word

The contents of data words DW 20 to DW 100 are to be set to logic 0. The "index register" for the parameter of the data words is DW 0.

```

 : L KF 20 Assignment of "index register"
 : T DW1
M1 : L KF 0 Reset
 : DO DW1
 : T DW0
 : L DW1 Increment the index register
 : L KF 1
 :+F
 : T DW1
 : L KF 100
 : <=F
 : JC =M1 Jump if index is in range
 Other STEP 5 program

```



**Example: Implementation of spindle override using a branch destination list**

Branch destination lists work with addresses (i.e. words), this means the jump displacement must be calculated from the instructions used.

In this example we have a jump displacement of two addresses because we have used two single-word instructions (L KBx and JU=M003).

Be careful if you use the two-word instruction L KMx instead of the single-word instruction L KBx. In this case you must use a jump displacement of 3.

NAME: SP-KORR

|      |        |      |       |                                  |
|------|--------|------|-------|----------------------------------|
|      | : AN   | M    | 3.1   | Default value of 50 % for        |
|      | : JC = | M001 |       | first execution of OB1           |
|      | : L    | KB1  |       |                                  |
|      | : S    | C    | 1     |                                  |
|      | : A    | F    | 0.1   | One flag                         |
|      | : S    | Q    | 100.4 | Spindle override active          |
| M001 | : A    | I    | 87.2  | Hand-held unit key 19            |
|      | : CU   | C    | 1     | Count up                         |
|      | : A    | I    | 87.3  | Hand-held unit key 20            |
|      | : CD   | C    | 1     | Count down                       |
|      | :      |      |       |                                  |
|      | : L    | C    | 1     | Upper limit Z1=15                |
|      | : L    | KB   | 15    |                                  |
|      | :>F    |      |       |                                  |
|      | : =    | F    | 187.0 |                                  |
|      | : U    | F    | 187.0 |                                  |
|      | : L    | KB   | 15    |                                  |
|      | : S    | C    | 1     |                                  |
|      | :      |      |       |                                  |
|      | : L    | C    | 1     | Lower limit Z1=1                 |
|      | : L    | KB   | 0     |                                  |
|      | : !=F  |      |       |                                  |
|      | : =    | F    | 187.1 |                                  |
|      | : U    | F    | 187.1 |                                  |
|      | : L    | KB   | 1     |                                  |
|      | : S    | C    | 1     |                                  |
|      | :      |      |       |                                  |
|      | : L    | C    | 1     | Double counter content,          |
|      | : L    | C    | 1     | because destination list         |
|      | : +F   |      |       | counts instructions,             |
|      | : T    | FW   | 190   | and there are 2                  |
|      | :      |      |       | instructions per jump            |
|      | :      |      |       |                                  |
|      | : DO   | FW   | 190   | Beginning of destination list    |
| M002 | : JU = | M002 |       |                                  |
|      | :      |      |       | Jump displacement 1 (Blank line) |
|      | : L    | KB   | 1     | Jump displacement 2              |
|      | : JU = | M003 |       |                                  |
|      | : L    | KB   | 3     | Jump displacement 4              |
|      | : JU = | M003 |       |                                  |
|      | : L    | KB   | 2     | Jump displacement 6              |
|      | : JU = | M003 |       |                                  |
|      | : L    | KB   | 6     | Jump displacement 8              |
|      | : JU = | M003 |       |                                  |

```

: L KB 7 Jump displacement 10
: JU = M003
: L KB 5 Jump displacement 12
: JU = M003
: L KB 4 Jump displacement 14
: JU = M003
: L KB 12 Jump displacement 16
: JU = M003
: L KB 13 Jump displacement 18
: JU = M003
: L KB 15 Jump displacement 20
: JU = M003
: L KB 14 Jump displacement 22
: JU = M003
: L KB 10 Jump displacement 24
: JU = M003
: L KB 11 Jump displacement 26
: JU = M003
: L KB 9 Jump displacement 28
: JU = M003
: L KB 8 Jump displacement 30
: JU = M003
:
M003 : T FY 100
: L QB 100 Save first 4 bits
: L KM 00000000 1111000 from QB100
: AW and function
: L FY 188 Add the spindle override
: OW to QB100; or function
: T QB 100
: BE

```

### 9.4.13 STEP 5 commands with direct memory access

| Operation           | Description                                                                             |
|---------------------|-----------------------------------------------------------------------------------------|
| <b>LIR 0</b>        | Load ACCU 1:<br>with the contents of the memory word addressed via Accu 1               |
| <b>TIR 2</b>        | Transfer ACCU 2:<br>to the memory word addressed via the contents of Accu 1             |
| <b>TNB 0 to 255</b> | Block transfer byte by byte, source address in Accu 2,<br>destination address in Accu 1 |
| <b>TNW 0 to 255</b> | Block transfer word by word, source address in Accu 2,<br>destination address in Accu 1 |

The LIR, TIR, TNB and TNW commands must first be enabled via machine data (MD 2003, bit 4 = 1) if they are to be used in the user program. Otherwise an error message is output and the PLC branches into a stop loop.

Only commands LIR 0 and TIR 2 will be executed. However the parameters will not be verified, i.e. any value can be programmed. No error message appears.

In the case of commands with direct memory access (LIR, TIR, TNB, TNW) an offset and a segment address are required. In this case the segment is specified by entering the segment number in the high word of the accumulator.

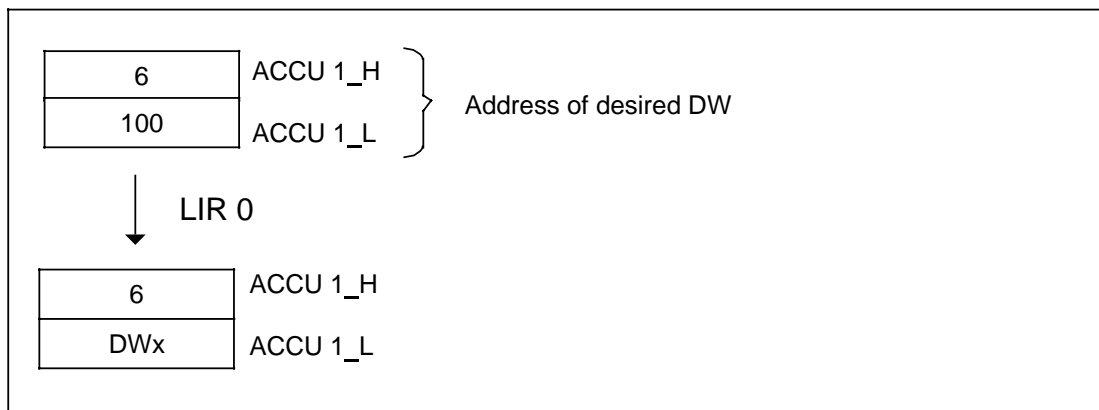
**The assignments are as follows:**

| Segment no. | Segment                                                       |
|-------------|---------------------------------------------------------------|
| 1           | PLC system data<br>(e.g. process images, block address lists) |
| 2           | Hardware<br>(e.g. EIA signal converter, MPC link memory)      |
| 3           | Internal NC-PLC interface                                     |
| 4           | Spindle interface                                             |
| 5           | User program memory                                           |
| 6           | User data memory                                              |
| 7           | PLC system program                                            |
| 8           | Resident function blocks (e.g. FB11)                          |

Segment number 1-10 are permitted for read access (source). Only segments 1-6 are permitted as destination for commands TIR, TNB and TNW. If other numbers are assigned the PLC goes into the stop state and an error message is output.

**Example 1: Read access to particular address**

L KB 6           Segment address for DWx ACCU 1\_H  
TFW 250  
L KH 0100       Offset address for DWx ACCU 1\_L  
TFW 252  
L FD 250  
LIR 0           Load DWx to ACCU 1\_L

**Example 2: Write operation to particular address**

L KB 6           Segment address for value x  
T FW 250  
L KH 0100       Offset address for value x  
T FW 252  
L KH FFFF       Load value x  
L FD 250  
TIR 2           Transfer value x user data segment

**9.4.14 Other operations**

| Operation                                                    | Description                                                                            |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <b>ADD BF -127 to +127</b><br><b>ADD KF -32768 to +32767</b> | Add byte constant (fixed-point) to Accu 1<br>Add fixed-point constant (word) to Accu 1 |
| <b>STS</b><br><b>TAK</b>                                     | Stop command<br>Swap contents of Accu 1 and Accu 2                                     |

When the instruction STS (stop immediately) is executed the programmable controller immediately enters the stop state. The current content of the accumulators can be evaluated via the U stack.

## 9.5 Overview of STEP 5 instructions

| Logic operations, binary (section : 9.3.1) |     |               |                               |                                                     |
|--------------------------------------------|-----|---------------|-------------------------------|-----------------------------------------------------|
| Operation                                  |     | Parameter     | Execution time (microseconds) | OPCODE                                              |
| A                                          | I   | 0.0 to 127.7  | 0.55                          | C 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | Q   | 0.0 to 127.7  | 0.55                          | C 0 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | F   | 0.0 to 255.7  | 0.55                          | 8 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | D   | 0.0 to 255.15 | 0.75                          | 783F 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| A                                          | N I | 0.0 to 127.7  | 0.55                          | E 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | N Q | 0.0 to 127.7  | 0.55                          | E 0 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | N F | 0.0 to 255.7  | 0.55                          | Q 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| A                                          | N D | 0.0 to 255.15 | 0.75                          | 783F 2 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| O                                          | I   | 0.0 to 127.7  | 0.55                          | C 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | Q   | 0.0 to 127.7  | 0.55                          | C 8 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | F   | 0.0 to 255.7  | 0.55                          | 8 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | D   | 0.0 to 255.15 | 0.75                          | 783F 1 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| O                                          | N I | 0.0 to 127.7  | 0.55                          | E 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | N Q | 0.0 to 127.7  | 0.55                          | E 8 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | N F | 0.0 to 255.7  | 0.55                          | A 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| O                                          | N D | 0.0 to 255.15 | 0.75                          | 783F 3 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| Q                                          | T   | 0 to 31       | 0.55                          | F 8 0 <sub>d</sub> 0 <sub>d</sub>                   |
| Q                                          | N T | 0 to 31       | 0.55                          | F C 0 <sub>d</sub> 0 <sub>d</sub>                   |
| Q                                          | C   | 0 to 31       | 0.55                          | 8 8 0 <sub>d</sub> 0 <sub>d</sub>                   |
| Q                                          | N C | 0 to 31       | 0.55                          | 8 C 0 <sub>d</sub> 0 <sub>d</sub>                   |
| O                                          | T   | 0 to 31       | 0.55                          | F 9 0 <sub>d</sub> 0 <sub>d</sub>                   |
| O                                          | N T | 0 to 31       | 0.55                          | F D 0 <sub>d</sub> 0 <sub>d</sub>                   |
| O                                          | C   | 0 to 31       | 0.55                          | 8 9 0 <sub>d</sub> 0 <sub>d</sub>                   |
| O                                          | N C | 0 to 31       | 0.55                          | 8 D 0 <sub>d</sub> 0 <sub>d</sub>                   |
| O                                          | (   |               | 0.55                          | B B 0 0                                             |
| Q                                          | (   |               | 0.55                          | B Q 0 0                                             |
| )                                          |     |               | 0.55                          | B F 0 0                                             |
| O                                          |     |               | 0.55                          | F B 0 0                                             |

See next page for explanation of indices

## Explanation of indices:

|   |                         |
|---|-------------------------|
| a | +Byte address           |
| b | +Bit address            |
| c | +Formal operand address |
| d | +Operand value          |
| e | +Constant               |
| f | +Block number           |
| g | +Word address           |
| h | +Shift number           |
| i | +Relative jump address  |
| k | +Register number        |
| l | +Block length           |
| m | +Jump width (16 bits)   |

| Store operations (section 9.3.2) |               |                               |                                                     |
|----------------------------------|---------------|-------------------------------|-----------------------------------------------------|
| Operation                        | Parameter     | Execution time (microseconds) | OPCODE                                              |
| S I                              | 0.0 to 127.7  | 0.8                           | D 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| S Q                              | 0.0 to 127.7  | 0.8                           | D 0 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| S F                              | 0.0 to 255.7  | 0.8                           | 9 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| S D                              | 0.0 to 255.15 | 1.0                           | 783F 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| R I                              | 0.0 to 127.7  | 0.8                           | F 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| R Q                              | 0.0 to 127.7  | 0.8                           | F 0 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| R F                              | 0.0 to 255.7  | 0.8                           | B 0 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| R D                              | 0.0 to 255.15 | 1.0                           | 783F 5 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| = I                              | 0.0 to 127.7  | 0.8                           | D 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| = Q                              | 0.0 to 127.7  | 0.8                           | D 8 <sub>b</sub> 8 <sub>a</sub> 0 <sub>a</sub>      |
| = F                              | 0.0 to 255.7  | 0.8                           | 9 8 <sub>b</sub> 0 <sub>a</sub> 0 <sub>a</sub>      |
| = D                              | 0.0 to 255.15 | 1.0                           | 783F 6 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |

| Load and transfer operations (section 9.3.3) |                               |                               |                                                                  |
|----------------------------------------------|-------------------------------|-------------------------------|------------------------------------------------------------------|
| Operation                                    | Parameter                     | Execution time (microseconds) | OPCODE                                                           |
| L IB                                         | 0 to 127                      | 0.5                           | 4 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L IW                                         | 0 to 126                      | 0.5/0.75 <sup>1)</sup>        | 5 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L ID                                         | 0 to 124                      | 0.75/1.0                      | 5 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L QB                                         | 0 to 127                      | 0.5                           | 5 2 8 <sub>d</sub> 0 <sub>d</sub>                                |
| L QW                                         | 0 to 126                      | 0.5/0.75                      | 5 2 8 <sub>d</sub> 0 <sub>d</sub>                                |
| L QD                                         | 0 to 124                      | 0.75/1.0                      | 5 A 8 <sub>d</sub> 0 <sub>d</sub>                                |
| L FB                                         | 0 to 255                      | 0.5                           | 0 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L FW                                         | 0 to 254                      | 0.5/0.75                      | 1 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L FD                                         | 0 to 252                      | 0.75/1.0                      | 1 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L DR                                         | 0 to 255                      | 0.5                           | 2 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L DL                                         | 0 to 255                      | 0.5                           | 2 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L DW                                         | 0 to 255                      | 0.5/0.75                      | 3 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L DD                                         | 0 to 254                      | 0.75/1.0                      | 3 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L T                                          | 0 to 31                       | 0.5                           | 0 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L C                                          | 0 to 31                       | 0.5                           | 4 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L PB                                         | 0 to 255                      | 1.0                           | 7 2 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L PW                                         | 0 to 126<br>128 to 254        | 1.0<br>1.75                   | 7 A 0 <sub>d</sub> 0 <sub>d</sub>                                |
| L KB                                         | 0 to 255                      | 0.5                           | 2 8 0 <sub>e</sub> 0 <sub>e</sub>                                |
| L KS                                         | any 2 alphanumeric characters | 0.5                           | 3010 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |
| L KT                                         | 0.1 to 999.3                  | 0.5                           | 3002 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |
| L KC                                         | 0 to 999                      | 0.5                           | 3001 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |
| L KM                                         | any bit pattern (16 bits)     | 0.5                           | 3080 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |
| L KH                                         | 0 to FFFF                     | 0.5                           | 3040 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |

1) The second value in execution time e.g. for L IW execution time 0.75/1.0 applies if this instruction was programmed at an odd address.

Examples : L IW0 = IB0 and IB1 even address  
 n-L IW1 = IB1 and IB2 odd address

| Load and transfer operations (section 9.3.3) |                                                                       |                               |                                                                                                                                 |
|----------------------------------------------|-----------------------------------------------------------------------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Operation                                    | Parameter                                                             | Execution time (microseconds) | OPCODE                                                                                                                          |
| L KF                                         | - 32768 to +32767                                                     | 0.5                           | 3004 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub>                                                                |
| L KG                                         | $\pm 0.1701412 \cdot 10^{39}$<br>to<br>$\pm 0.1469368 \cdot 10^{-38}$ | 1.0                           | 3800 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub><br>0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |
| L KY                                         | 0 to 255 for each byte                                                | 0.5                           | 3020 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub>                                                                |
| T IB                                         | 0 to 127                                                              | 0.5                           | 4 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T IW                                         | 0 to 126                                                              | 0.5/0.75                      | 5 3 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T ID                                         | 0 to 124                                                              | 0.75/1.0                      | 5 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T QB                                         | 0 to 127                                                              | 0.5                           | 4 B 8 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T QW                                         | 0 to 126                                                              | 0.5/0.75                      | 5 3 8 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T QD                                         | 0 to 124                                                              | 0.75/1.0                      | 5 B 8 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T FY                                         | 0 to 255                                                              | 0.5                           | 0 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T FW                                         | 0 to 254                                                              | 0.5/0.75                      | 1 3 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T FD                                         | 0 to 252                                                              | 0.75/1.0                      | 1 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T DR                                         | 0 to 255                                                              | 0.5                           | 2 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T DL                                         | 0 to 255                                                              | 0.5                           | 2 3 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T DW                                         | 0 to 255                                                              | 0.5/0.75                      | 3 3 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T DD                                         | 0 to 254                                                              | 0.75/1.0                      | 3 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T PB                                         | 0 to 127<br>128 to 255                                                | 1.0                           | 7 3 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |
| T PW                                         | 0 to 126<br>128 to 254                                                | 1.0<br>1.75                   | 7 B 0 <sub>d</sub> 0 <sub>d</sub>                                                                                               |



| Timer and counter operations (section 9.3.4) |   |           |                               |        |   |                               |
|----------------------------------------------|---|-----------|-------------------------------|--------|---|-------------------------------|
| Operation                                    |   | Parameter | Execution time (microseconds) | OPCODE |   |                               |
| SP                                           | T | 0 to 31   |                               | 3      | 4 | 0 <sub>d</sub> 0 <sub>d</sub> |
| SE                                           | T | 0 to 31   |                               | 1      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| SD                                           | T | 0 to 31   |                               | 2      | 4 | 0 <sub>d</sub> 0 <sub>d</sub> |
| SS                                           | T | 0 to 31   |                               | 2      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| SF                                           | T | 0 to 31   |                               | 1      | 4 | 0 <sub>d</sub> 0 <sub>d</sub> |
| R                                            | T | 0 to 31   |                               | 3      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| S                                            | C | 0 to 31   |                               | 5      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| R                                            | C | 0 to 31   |                               | 7      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| CU                                           | C | 0 to 31   |                               | 6      | C | 0 <sub>d</sub> 0 <sub>d</sub> |
| CD                                           | C | 0 to 31   |                               | 5      | 4 | 0 <sub>d</sub> 0 <sub>d</sub> |

| Compare operations (section 9.3.5) |  |           |                               |        |   |     |
|------------------------------------|--|-----------|-------------------------------|--------|---|-----|
| Operation                          |  | Parameter | Execution time (microseconds) | OPCODE |   |     |
| !=F                                |  |           | 0.25                          | 2      | 1 | 8 0 |
| ><F                                |  |           | 0.25                          | 2      | 1 | 6 0 |
| >F                                 |  |           | 0.25                          | 2      | 1 | 2 0 |
| >=F                                |  |           | 0.25                          | 2      | 1 | Q 0 |
| <F                                 |  |           | 0.25                          | 2      | 1 | 4 0 |
| <=F                                |  |           | 0.25                          | 2      | 1 | C 0 |
| !=D                                |  |           | 0.25                          | 3      | 9 | 8 0 |
| ><D                                |  |           | 0.25                          | 3      | 9 | 6 0 |
| >D                                 |  |           | 0.25                          | 3      | 9 | 2 0 |
| >=D                                |  |           | 0.25                          | 3      | 9 | Q 0 |
| <D                                 |  |           | 0.25                          | 3      | 9 | 4 0 |
| <=D                                |  |           | 0.25                          | 3      | 9 | C 0 |

The timers T16 to T31 must be enabled via PLC MD6.

| <b>Block calls (section 9.3.6)</b> |           |                               |                                   |
|------------------------------------|-----------|-------------------------------|-----------------------------------|
| Operation                          | Parameter | Execution time (microseconds) | OPCODE                            |
| JU PB                              | 0 to 255  | 2.0                           | 7 5 0 <sub>f</sub> 0 <sub>f</sub> |
| JU FB                              | 0 to 255  | 2.0                           | 3 D 0 <sub>f</sub> 0 <sub>f</sub> |
| JU SB                              | 0 to 255  | 2.0                           | 7 D 0 <sub>f</sub> 0 <sub>f</sub> |
| JC PB                              | 0 to 255  | 2.0                           | 5 5 0 <sub>f</sub> 0 <sub>f</sub> |
| JC FB                              | 0 to 255  | 2.0                           | 1 D 0 <sub>f</sub> 0 <sub>f</sub> |
| JC SB                              | 0 to 255  | 2.0                           | 5 D 0 <sub>f</sub> 0 <sub>f</sub> |
| Q DB                               | 0 to 255  | 1.0                           | 2 0 0 <sub>f</sub> 0 <sub>f</sub> |
| BI                                 |           | 2.0 / 2.25*                   | 6 5 0 0                           |
| BEC                                |           | 2.0 / 2.25*                   | 0 5 0 0                           |
| BEA                                |           | 2.0 / 2.25*                   | 6 5 0 1                           |

| <b>Coding operations (section 9.3.7)</b> |           |                               |                                   |
|------------------------------------------|-----------|-------------------------------|-----------------------------------|
| Operation                                | Parameter | Execution time (microseconds) | OPCODE                            |
| LC T                                     | 0 to 31   | 30.4-79.0                     | 0 C 0 <sub>d</sub> 0 <sub>d</sub> |
| LC C                                     | 0 to 31   | 28.0-76.6                     | 4 C 0 <sub>d</sub> 0 <sub>d</sub> |

| <b>Arithmetic operations (section 9.3.8)</b> |           |                               |         |
|----------------------------------------------|-----------|-------------------------------|---------|
| Operation                                    | Parameter | Execution time (microseconds) | OPCODE  |
| +F                                           |           | 0.25                          | 7 9 0 0 |
| - F                                          |           | 0.25                          | 4 9 0 0 |

---

\* If there is no reset to WOP!

| <b>Other operations (section 9.3.9)</b> |           |                               |         |
|-----------------------------------------|-----------|-------------------------------|---------|
| Operation                               | Parameter | Execution time (microseconds) | OPCODE  |
| NOP 0                                   |           | 0.6                           | 0 0 0 0 |
| NOP 1                                   |           | 0.6                           | F F F F |
| BLD 255                                 |           | 0.6                           | 1 0 F F |
| STP                                     |           |                               | 7 0 0 3 |

**Supplementary operations (only for FBs)**

| <b>Logic operations, binary (substituted) (section 9.4.1)</b> |                |                               |                                   |
|---------------------------------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                                                     | Parameter      | Execution time (microseconds) | OPCODE                            |
| A =                                                           | Formal operand | 0.75 <sup>1)</sup>            | 0 7 0 <sub>c</sub> 0 <sub>c</sub> |
| AN =                                                          | Formal operand | 0.75 <sup>1)</sup>            | 2 7 0 <sub>c</sub> 0 <sub>c</sub> |
| O =                                                           | Formal operand | 0.75 <sup>1)</sup>            | 0 F 0 <sub>c</sub> 0 <sub>c</sub> |
| ON =                                                          | Formal operand | 0.75 <sup>1)</sup>            | 2 F 0 <sub>c</sub> 0 <sub>c</sub> |

| <b>Setting operations (substituted) (section 9.4.2)</b> |                |                               |                                   |
|---------------------------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                                               | Parameter      | Execution time (microseconds) | OPCODE                            |
| S =                                                     | Formal operand | 0.75 <sup>1)</sup>            | 1 7 0 <sub>c</sub> 0 <sub>c</sub> |
| RB =                                                    | Formal operand | 0.75 <sup>1)</sup>            | 3 7 0 <sub>c</sub> 0 <sub>c</sub> |
| = =                                                     | Formal operand | 0.75 <sup>1)</sup>            | 1 F 0 <sub>c</sub> 0 <sub>c</sub> |

<sup>1)</sup>+ Time for substituted operation

| <b>Timer and counter operations (section 9.4.3)</b> |                |                               |                                   |
|-----------------------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                                           | Parameter      | Execution time (microseconds) | OPCODE                            |
| SP =                                                | Formal operand | 0.75 1)                       | 3 6 0 <sub>c</sub> 0 <sub>c</sub> |
| SR =                                                | Formal operand | 0.75 1)                       | 2 6 0 <sub>c</sub> 0 <sub>c</sub> |
| SEC =                                               | Formal operand | 0.75 1)                       | 1 E 0 <sub>c</sub> 0 <sub>c</sub> |
| SSU =                                               | Formal operand | 0.75 1)                       | 2 E 0 <sub>c</sub> 0 <sub>c</sub> |
| SFD =                                               | Formal operand | 0.75 1)                       | 1 6 0 <sub>c</sub> 0 <sub>c</sub> |
| RD =                                                | Formal operand | 0.75 1)                       | 3 E 0 <sub>c</sub> 0 <sub>c</sub> |

| <b>Enable operations for timer and counter operations (section 9.4.4)</b> |                |                               |                                   |
|---------------------------------------------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                                                                 | Parameter      | Execution time (microseconds) | OPCODE                            |
| FR T                                                                      | 0 to 31        |                               | 0 4 0 <sub>d</sub> 0 <sub>d</sub> |
| FR C                                                                      | 0 to 31        |                               | 4 4 0 <sub>d</sub> 0 <sub>d</sub> |
| FR =                                                                      | Formal operand | 0.75                          | 0 6 0 <sub>c</sub> 0 <sub>c</sub> |

---

1) + Time for substituted operation

| Bit test operations (section 9.4.5) |               |                               |                                                     |
|-------------------------------------|---------------|-------------------------------|-----------------------------------------------------|
| Operation                           | Parameter     | Execution time (microseconds) | OPCODE                                              |
| P I                                 | 0.0 to 127.7  | 0.75                          | 7038 C 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| P Q                                 | 0.0 to 127.7  | 0.75                          | 7038 C 0 <sub>b</sub> 8 <sub>d</sub> 0 <sub>d</sub> |
| P F                                 | 0.0 to 255.7  | 0.75                          | 7049 C 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| P C                                 | 0.0 to 31.15  | 0.75                          | 7015 C 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| P T                                 | 0.0 to 31.15  | 0.75                          | 7025 C 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| P D                                 | 0.0 to 255.15 | 0.75                          | 7046 C 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| TBN I                               | 0.0 to 127.7  | 0.75                          | 7038 8 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| TBN Q                               | 0.0 to 127.7  | 0.75                          | 7038 8 0 <sub>b</sub> 8 <sub>d</sub> 0 <sub>d</sub> |
| TBN F                               | 0.0 to 255.7  | 0.75                          | 7049 8 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| TBN C                               | 0.0 to 31.15  | 0.75                          | 7015 8 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| TBN T                               | 0.0 to 31.15  | 0.75                          | 7025 8 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| TBN D                               | 0.0 to 255.15 | 0.75                          | 7046 8 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| SU I                                | 0.0 to 127.7  | 1.0                           | 7038 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| SU Q                                | 0.0 to 127.7  | 1.0                           | 7038 4 0 <sub>b</sub> 8 <sub>d</sub> 0 <sub>d</sub> |
| SU F                                | 0.0 to 255.7  | 1.0                           | 7049 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| SU C                                | 0.0 to 31.15  | 1.0                           | 7015 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| SU T                                | 0.0 to 31.15  | 1.0                           | 7025 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| SU D                                | 0.0 to 255.15 | 1.0                           | 7046 4 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| RU I                                | 0.0 to 127.7  | 1.0                           | 7038 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| RU Q                                | 0.0 to 127.7  | 1.0                           | 7038 0 0 <sub>b</sub> 8 <sub>d</sub> 0 <sub>d</sub> |
| RU F                                | 0.0 to 255.7  | 1.0                           | 7049 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| RU C                                | 0.0 to 31.15  | 1.0                           | 7015 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| RU T                                | 0.0 to 31.15  | 1.0                           | 7025 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |
| RU D                                | 0.0 to 255.15 | 1.0                           | 7046 0 0 <sub>b</sub> 0 <sub>d</sub> 0 <sub>d</sub> |

| <b>Coding, load and transfer operations (section 9.4.6)</b> |                |                               |                                   |
|-------------------------------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                                                   | Parameter      | Execution time (microseconds) | OPCODE                            |
| L =                                                         | Formal operand | 0.75 <sup>1)</sup>            | 4 6 0 <sub>c</sub> 0 <sub>c</sub> |
| LC =                                                        | Formal operand | 0.75 <sup>1)</sup>            | 0 E 0 <sub>c</sub> 0 <sub>c</sub> |
| LW =                                                        | Formal operand | 0.5                           | 3 F 0 <sub>c</sub> 0 <sub>c</sub> |
| LD =                                                        | Formal operand | 0.75                          | 5 6 0 <sub>c</sub> 0 <sub>c</sub> |
| T =                                                         | Formal operand | 0.75 <sup>1)</sup>            | 6 6 0 <sub>c</sub> 0 <sub>c</sub> |

| <b>Logic operations, digital (section 9.4.7)</b> |           |                               |         |
|--------------------------------------------------|-----------|-------------------------------|---------|
| Operation                                        | Parameter | Execution time (microseconds) | OPCODE  |
| AW                                               |           | 0.25                          | 4 1 0 0 |
| OW                                               |           | 0.25                          | 4 9 0 0 |
| XOW                                              |           | 0.25                          | 5 1 0 0 |

| <b>Shift and rotate instructions (section 9.4.8)</b> |           |                               |                                   |
|------------------------------------------------------|-----------|-------------------------------|-----------------------------------|
| Operation                                            | Parameter | Execution time (microseconds) | OPCODE                            |
| SLW                                                  | 0 to 15   | 0.5 - 1.5                     | 6 1 0 0 <sub>h</sub>              |
| SRW                                                  | 0 to 15   | 0.5 - 1.5                     | 6 9 0 0 <sub>h</sub>              |
| SVW                                                  | 0 to 15   | 0.5 - 1.5                     | 6 8 0 <sub>h</sub> 1              |
| SVD                                                  | 0 to 32   | 0.5 - 2.5                     | 7 1 0 <sub>h</sub> 0 <sub>h</sub> |
| SLD                                                  | 0 to 32   | 0.5 - 2.5                     | 2 9 0 <sub>h</sub> 0 <sub>h</sub> |

<sup>1)</sup>+ Time for substituted operation

| <b>Conversion operations (section 9.4.9)</b> |           |                               |         |
|----------------------------------------------|-----------|-------------------------------|---------|
| Operation                                    | Parameter | Execution time (microseconds) | OPCODE  |
| CFW                                          |           | 0.25                          | 0 1 0 0 |
| CSW                                          |           | 0.25                          | 0 9 0 0 |

| <b>Decrement and increment (section 9.4.10)</b> |           |                               |         |
|-------------------------------------------------|-----------|-------------------------------|---------|
| Operation                                       | Parameter | Execution time (microseconds) | OPCODE  |
| D                                               | 0 to 255  | 0.25                          | 1 9 0 0 |
| I                                               | 0 to 255  | 0.25                          | 1 1 0 0 |

| <b>Jump operations (section 9.4.11)</b> |                                     |                               |                                              |
|-----------------------------------------|-------------------------------------|-------------------------------|----------------------------------------------|
| Operation                               | Parameter                           | Execution time (microseconds) | OPCODE                                       |
| JU=                                     | Symbolic address up to 4 characters | 0.25                          | Z D 0 <sub>i</sub> 0 <sub>i</sub>            |
| JC=                                     | Symbolic address up to 4 characters | 0.25                          | F A 0 <sub>i</sub> 0 <sub>i</sub>            |
| JZ=                                     | Symbolic address up to 4 characters | 0.25                          | 4 5 0 <sub>i</sub> 0 <sub>i</sub>            |
| JN=                                     | Symbolic address up to 4 characters | 0.25                          | 3 5 0 <sub>i</sub> 0 <sub>i</sub>            |
| JP=                                     | Symbolic address up to 4 characters | 0.25                          | 1 5 0 <sub>i</sub> 0 <sub>i</sub>            |
| JM=                                     | Symbolic address up to 4 characters | 0.25                          | 2 5 0 <sub>i</sub> 0 <sub>i</sub>            |
| JO=                                     | Symbolic address up to 4 characters | 0.25                          | 0 D 0 <sub>i</sub> 0 <sub>i</sub>            |
| JS=                                     | Symbolic address up to 4 characters | 0.5                           | 6 0 0 C<br>0 0 0 <sub>i</sub> 0 <sub>i</sub> |

| Processing operations (section 9.4.12) |                |                               |                                   |
|----------------------------------------|----------------|-------------------------------|-----------------------------------|
| Operation                              | Parameter      | Execution time (microseconds) | OPCODE                            |
| B DW                                   | 0 to 255       | 0.5 <sup>1)</sup>             | 6 E 0 <sub>d</sub> 0 <sub>d</sub> |
| B MW                                   | 0 to 254       | 0.75 <sup>1)</sup>            | 4 E 0 <sub>d</sub> 0 <sub>d</sub> |
| B =                                    | Formal operand |                               | 7 6 0 0                           |

| STEP 5 instructions with direct memory access (section 9.4.13) |           |                               |                                   |
|----------------------------------------------------------------|-----------|-------------------------------|-----------------------------------|
| Operation                                                      | Parameter | Execution time (microseconds) | OPCODE                            |
| LIR                                                            | 0         |                               | 4 0 0 0 <sub>k</sub>              |
| TIR                                                            | 2         |                               | 4 8 0 0 <sub>k</sub>              |
| TNB                                                            | 0 to 255  |                               | 0 3 0 <sub>l</sub> 0 <sub>l</sub> |
| TNW                                                            | 0 to 255  |                               | 4 3 0 <sub>l</sub> 0 <sub>l</sub> |

| Other operations (section 9.4.14) |                    |                               |                                                                  |
|-----------------------------------|--------------------|-------------------------------|------------------------------------------------------------------|
| Operation                         | Parameter          | Execution time (microseconds) | OPCODE                                                           |
| ADD BF                            | - 128 to + 127     | 0.25                          | 5 0 0 <sub>d</sub> 0 <sub>d</sub>                                |
| ADD KF                            | - 32768 to + 32767 | 0.25                          | 5800 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> 0 <sub>e</sub> |

<sup>1)</sup>+ Time for substituted operation



# 10 Rules of compatibility between the LAD, CSF and STL methods of representation

## 10.1 General

Each method of representation in the STEP 5 programming language contains limits. It therefore follows that a program block written in an STL may not be output in an LAD or CSF without restrictions; similarly, the LAD and CSF which are both graphic methods of representation, may not be fully compatible. If the program was entered as a LAD or CSF, it can be retranslated into an STL.

The purpose of this section is to provide some rules which, when observed, ensure full compatibility between the three methods of representation.

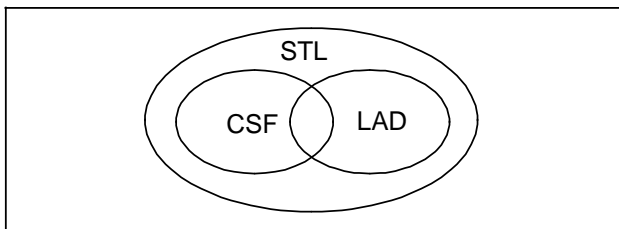
These rules are arranged as follows:

- Rules of compatibility for graphic program input (LAD, CSF)

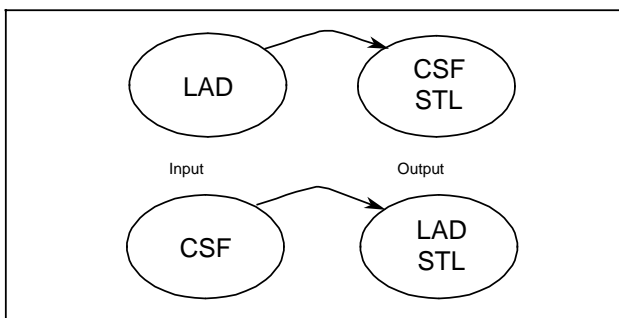
With input in a graphic method of representation, the observance of these rules allows output in the other methods of representation.

- Rules of compatibility for program input in a statement list

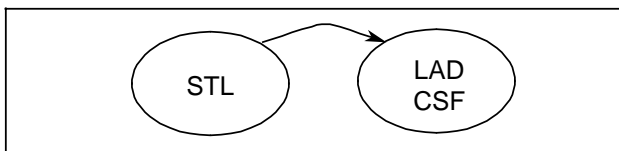
With input in the form of a statement list, the observance of these rules ensures output in the other methods of representation.



*Extent and restrictions of the methods of representation in the STEP 5 programming language*



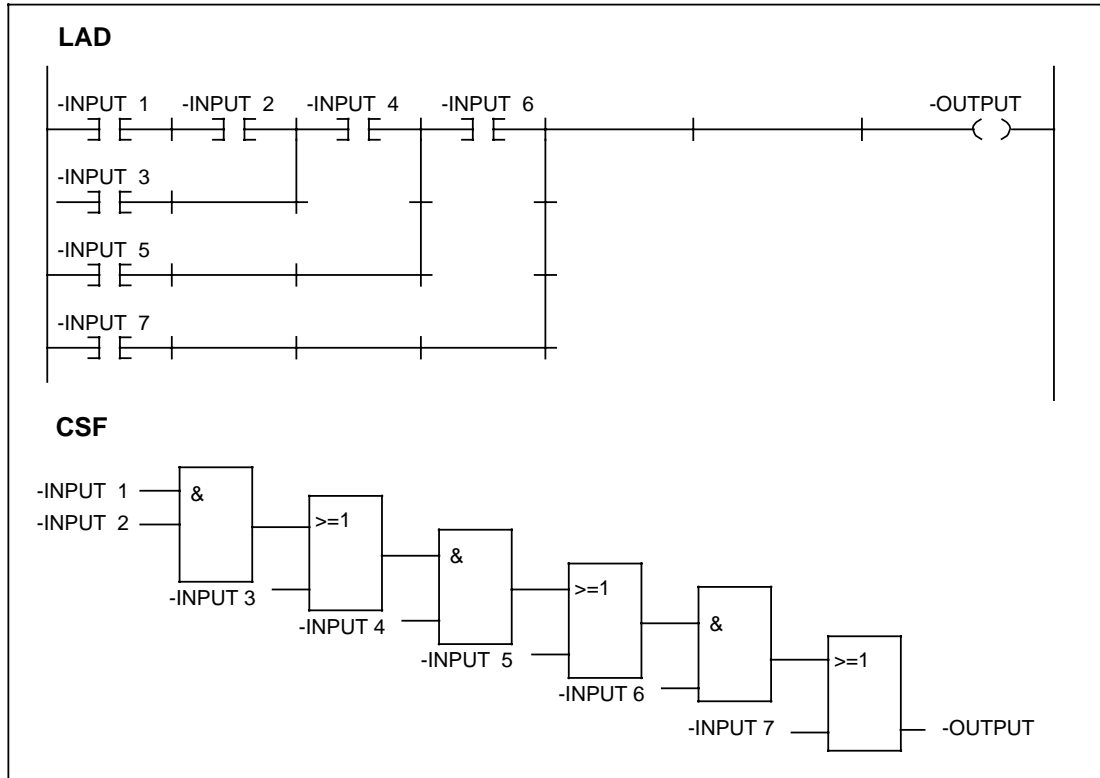
*Graphic input*



*Input in statement list*

## 10.2 Rules of compatibility for graphic program input (LAD, CSF)

Excessively deep nesting can result in the display limits (8 levels) being exceeded in the CSF.

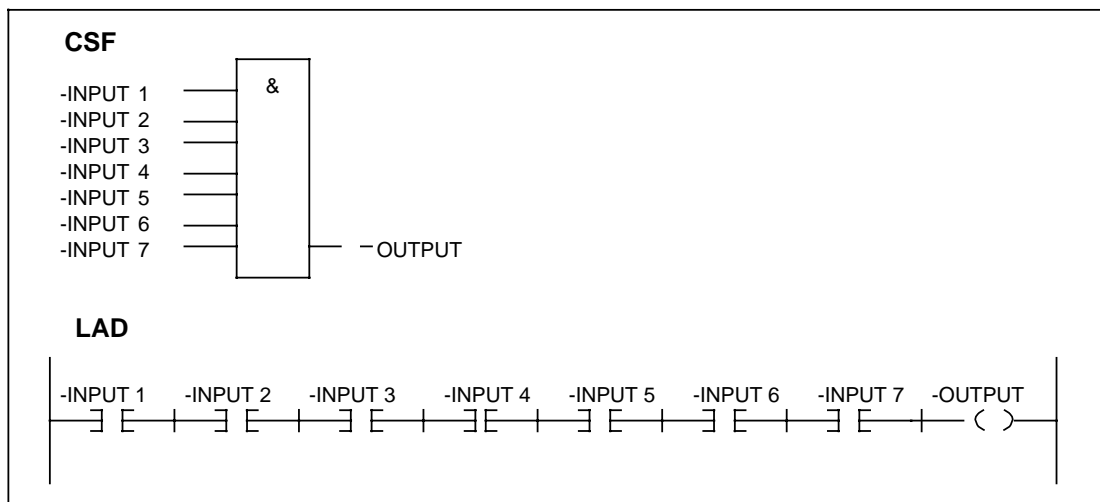


Example of maximum LAD nesting for output in CSF

### Input in CSF: Output in LAD and STL

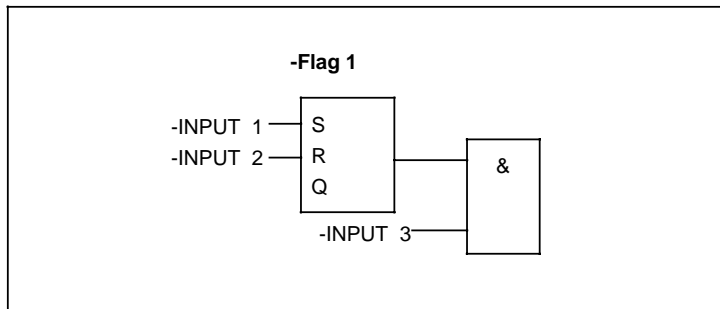
**Rule 1: Do not exceed the display limits for LAD:**

An excessive number of inputs at an CSF box results in exceeding of the LAD display limit.



Example of maximum AND-box expansion for output in LAD

**Rule 2:** *The output of a complex element (storage element, comparator, timer or counter) must not be ORed.*



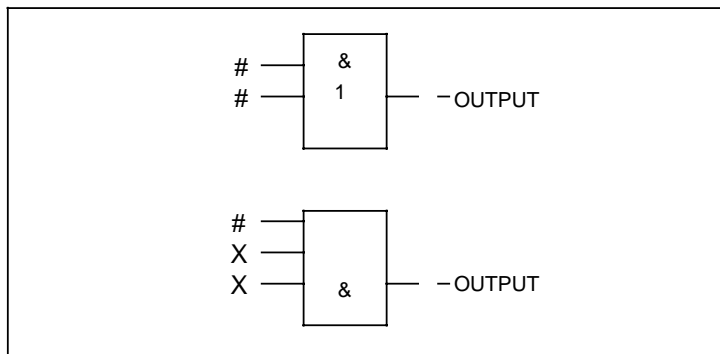
*Only AND-boxes are allowed following a complex element*

**Rule 3: Connectors**

- Connectors are always allowed with an OR-box.
- Connectors are only allowed at the first input with an AND-box.

(Connectors are intermediate flags which are used for economy with recurring logic operations).

# Connector allowed  
X Connector not allowed



*Example showing where connectors are allowed with OR and AND-boxes*

### 10.3 Rules of compatibility for program input in a statement list

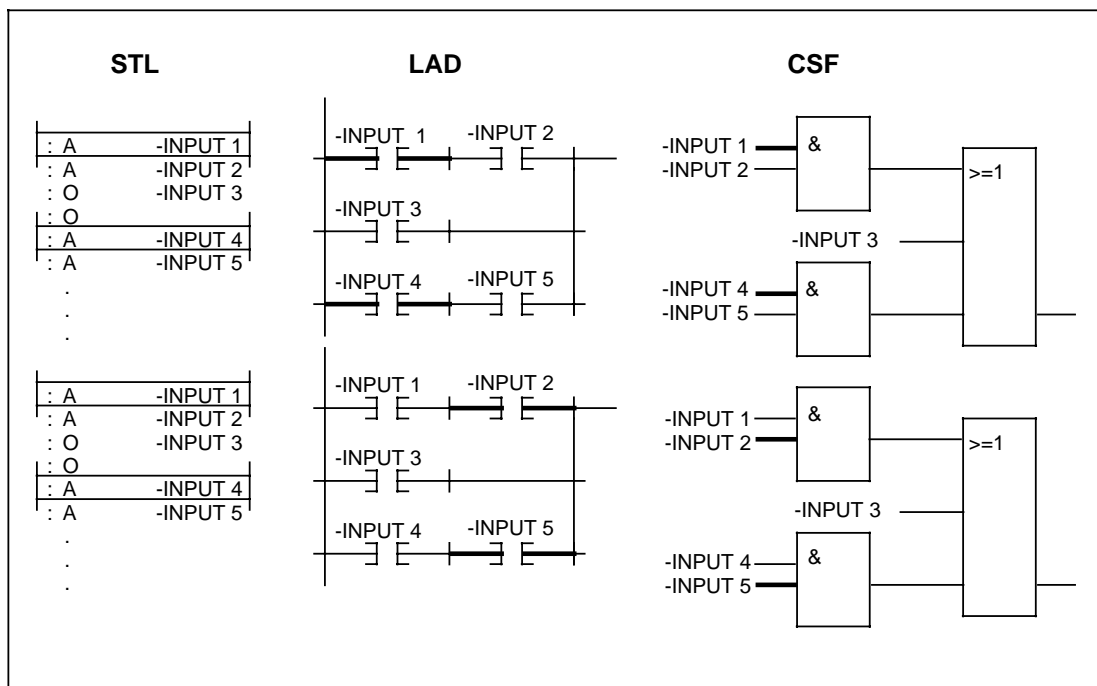
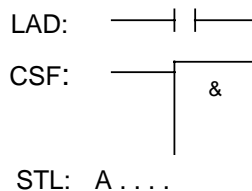
**Rule 1: AND operation:**

(Test of logic state and AND logic).

LAD: Contact in series

CSF: Input to an AND-box

STL: Statement A . . .



Explanations of the rule for AND operations

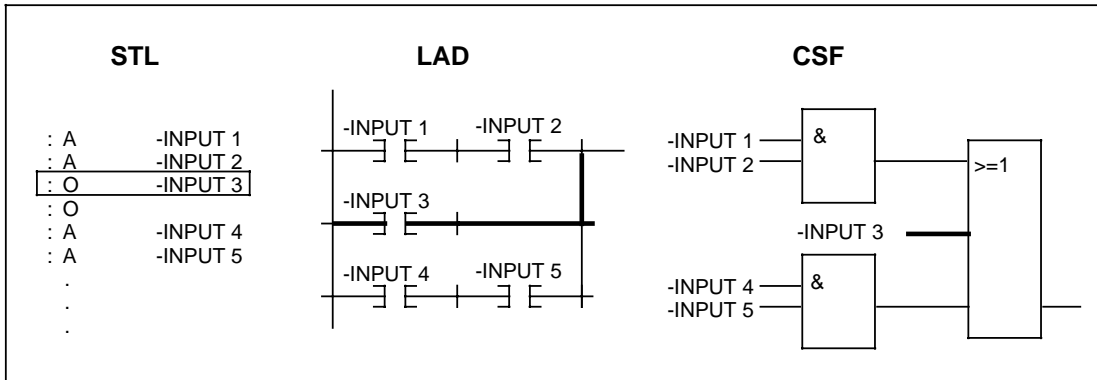
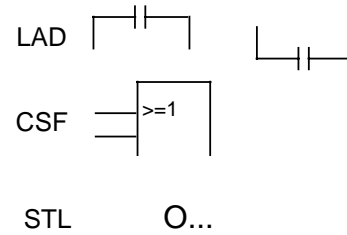
**Rule 2: OR operation**

(Test of logic state and OR logic).

LAD: Only one contact in a parallel branch

CSF: Input to an OR-box

STL: Statement O . . .



Explanations of the rule for OR operations

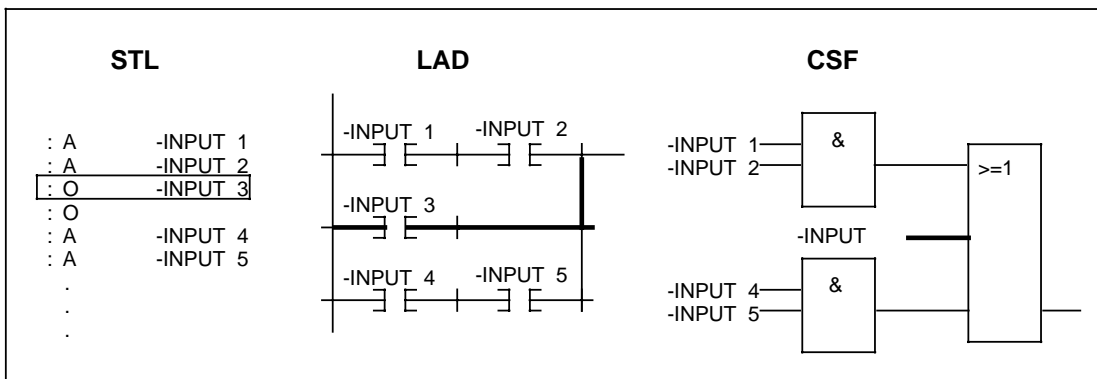
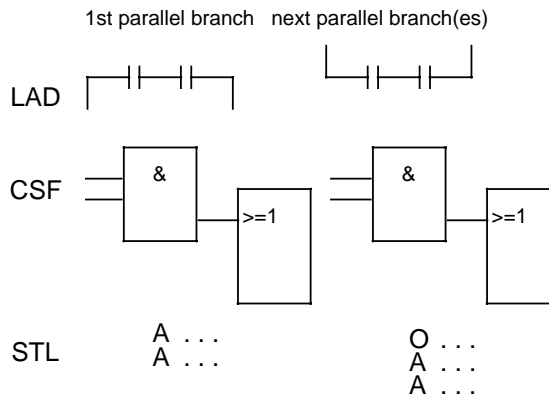
**Rule 3: AND before OR operation**

(OR operation before AND functions)

LAD: Two or more contacts in a parallel branch

CSF: AND-box before OR-box

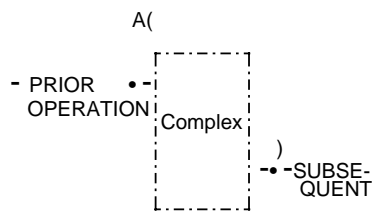
STL: Statements  $\left. \begin{matrix} O \dots \\ A \dots \\ A \dots \end{matrix} \right\}$



Explanations of the rule for AND before OR operation

**Rule 4: Parentheses**

Covered in this rule are the parenthesized, complex, enclosed binary operations or complex elements with prior or subsequent operations.

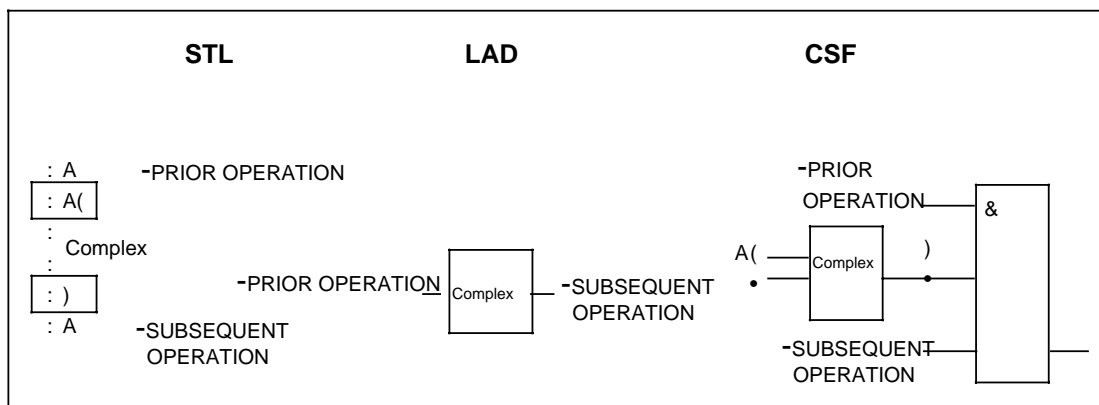


a) Complex binary operation

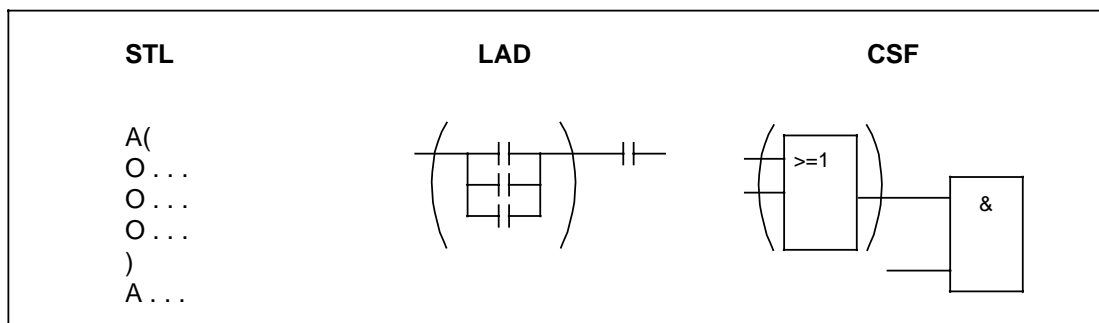
This class of operation includes the OR before AND operations, the rules for which are as follows:

- AND operation before OR functions  
 LAD : Sequencing of parallel contacts in series  
 CSF : OR-box before AND-box  
 STL : Statements A(  
                                   OR OPERATION  
                                   )  
                                   :  
                                   :

The OR before AND operations are a subset of the complex binary operations in which two parallel contacts form the simplest operation.



Explanations of parenthesized, complex binary functions

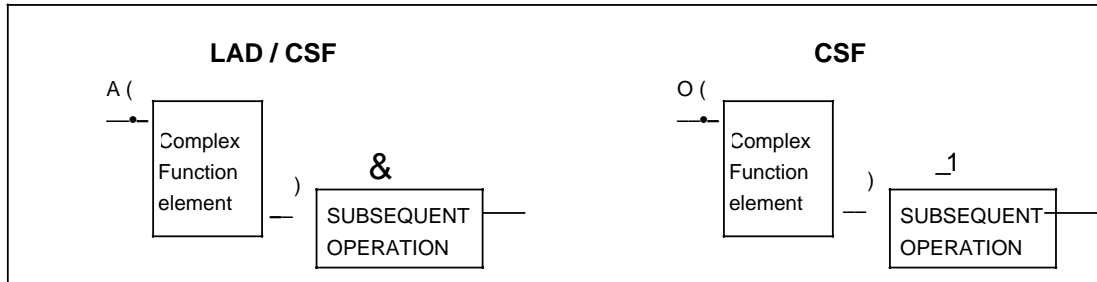


Explanations of the rule for OR before AND operation

## b) Complex elements (storage, timing, comparison and counting functions)

The following rules must be observed for complex elements:

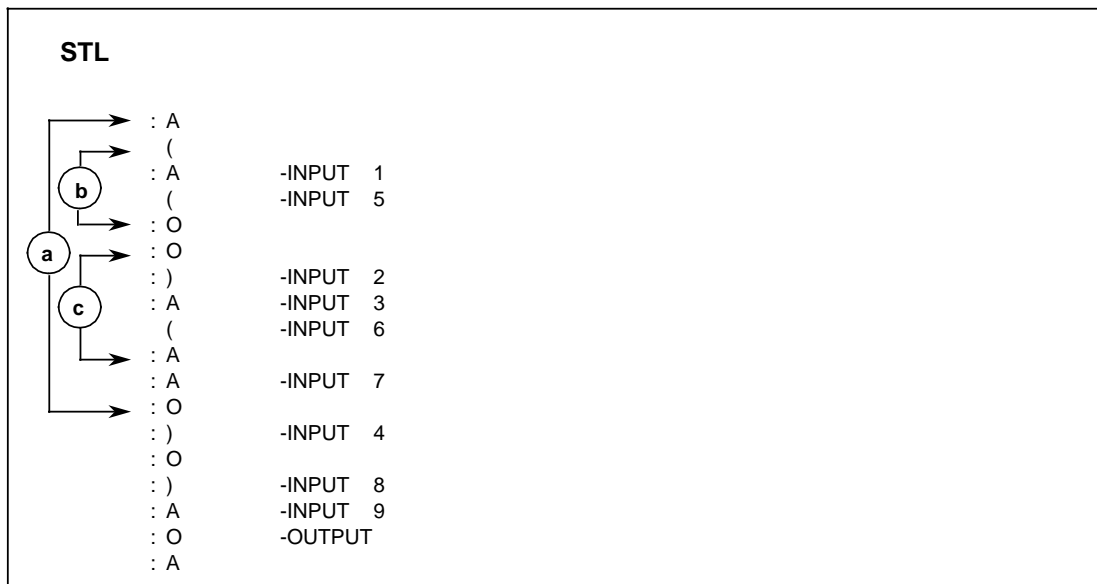
- No subsequent operation, no parentheses
- Subsequent operation AND: A ( . . . ) . . .
- Subsequent operation OR: O ( . . . ) . . . (only for CSF, not allowed for LAD)
- A complex element cannot have prior operations.



*Explanations of parentheses for complex elements*

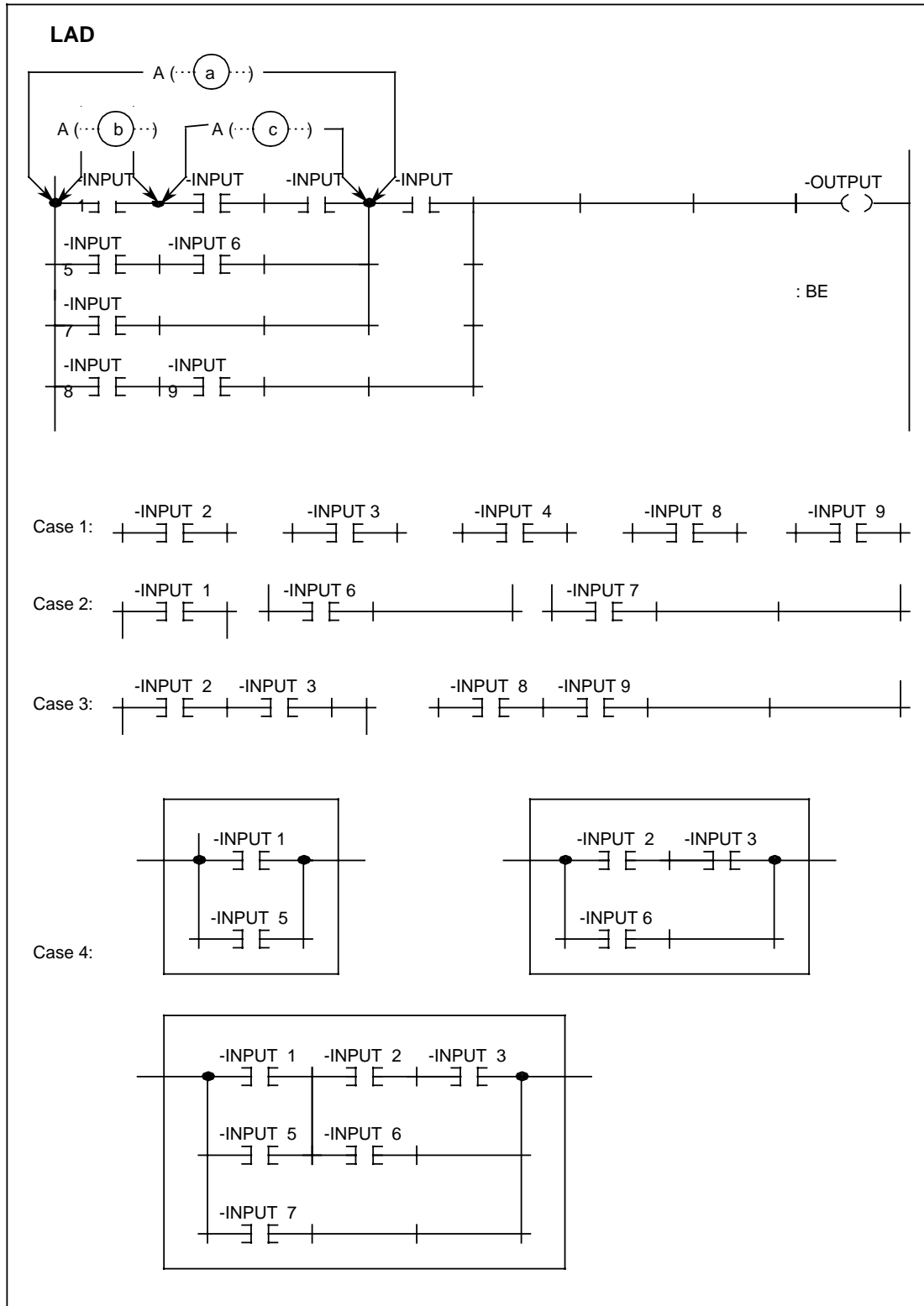
**Example 1:** LAD/STL

- Case 1 : AND (contact in series)  
 Case 2 : OR ( only 1 contact in a parallel branch)  
 Case 3 : AND before OR ( two or more contacts in a parallel branch)  
 Case 4 : OR before AND (parentheses )



*Example 1: LAD/STL (continued on next page)*

10.3 Rules of compatibility for program input in a statement list



Example 1: LAD/STL (continued)



NOP 0 must be applied to each unused input or output.

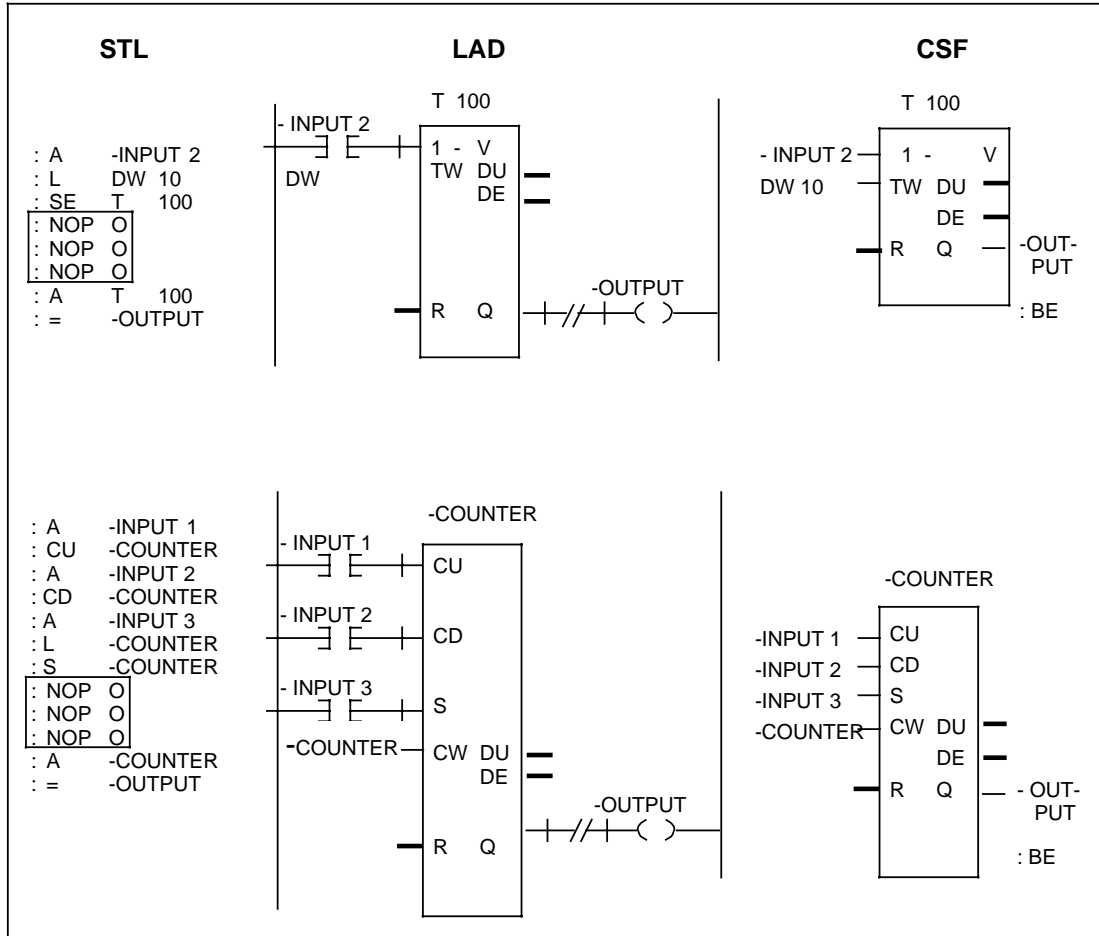
**Exception:**

S and TW for timers, and S and CW for counters must always be used jointly.

For STL programming, the complex elements must be programmed in the same order as the parameter assignment on the screen in the graphic method of representation.

**Exception:**

Time and count; the corresponding value must first be stored in the accumulator by a load command.



Example of assignments for unused inputs and outputs

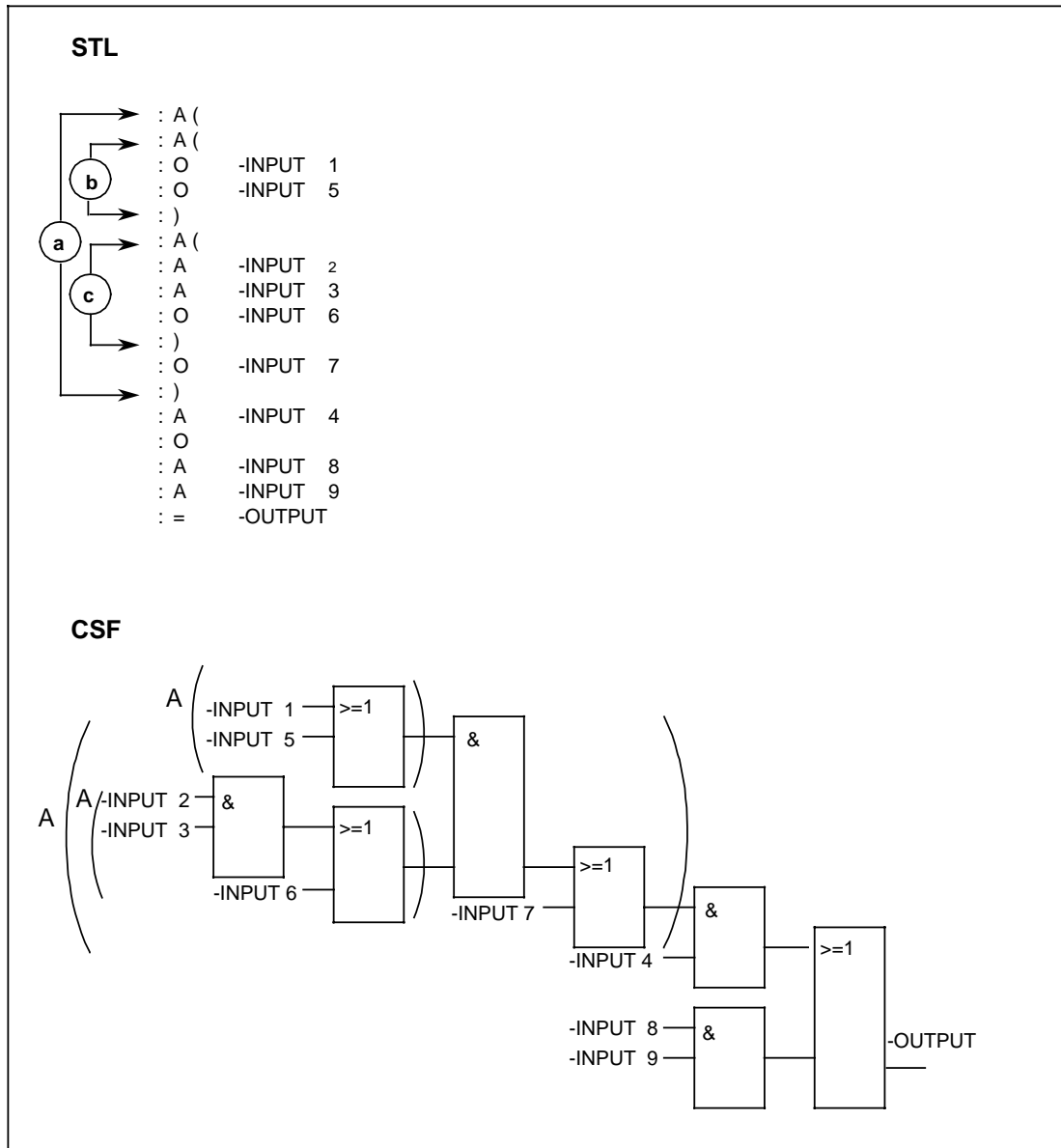
**Important note:**

Only one complex function element is allowed per network.

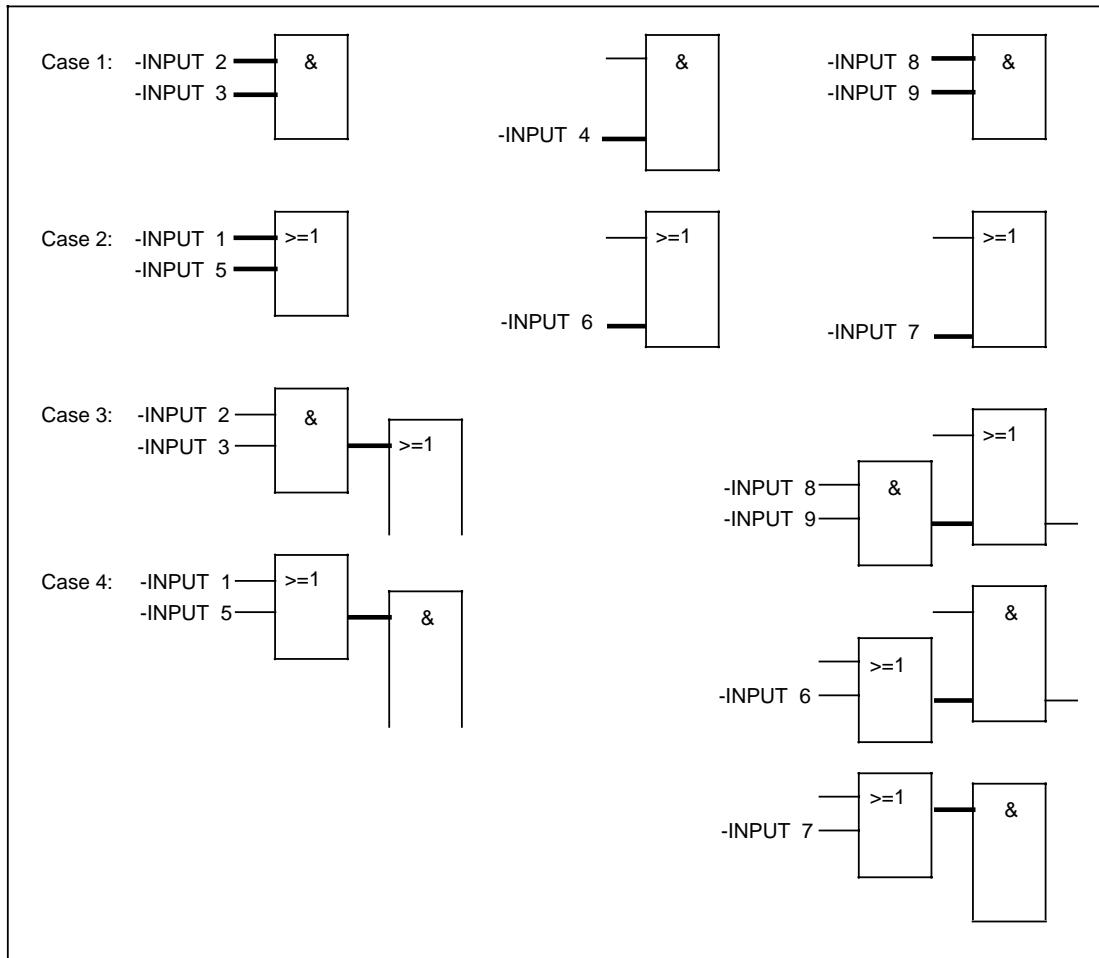
The following examples show the four cases presented in a complex binary operation: In the LAD and STL methods of representation and in the CSF and STL methods of representation.

**Example 2: CSF/STL**

- Case 1 : AND (input to an AND-box)
- Case 2 : OR (input to an OR-box)
- Case 3 : AND before OR (AND-box before OR-box)
- Case 4 : OR before AND (OR-box before AND-box)



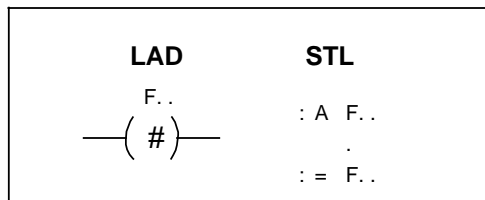
Example 2: CSF/STL (continued on next page)



Example 2: CSF/STL (continued)

**Rule 5: Connectors**

For the sake of clarity, the rules for connectors are listed separately for the LAD and LOD methods of representation. The following example is given for both.

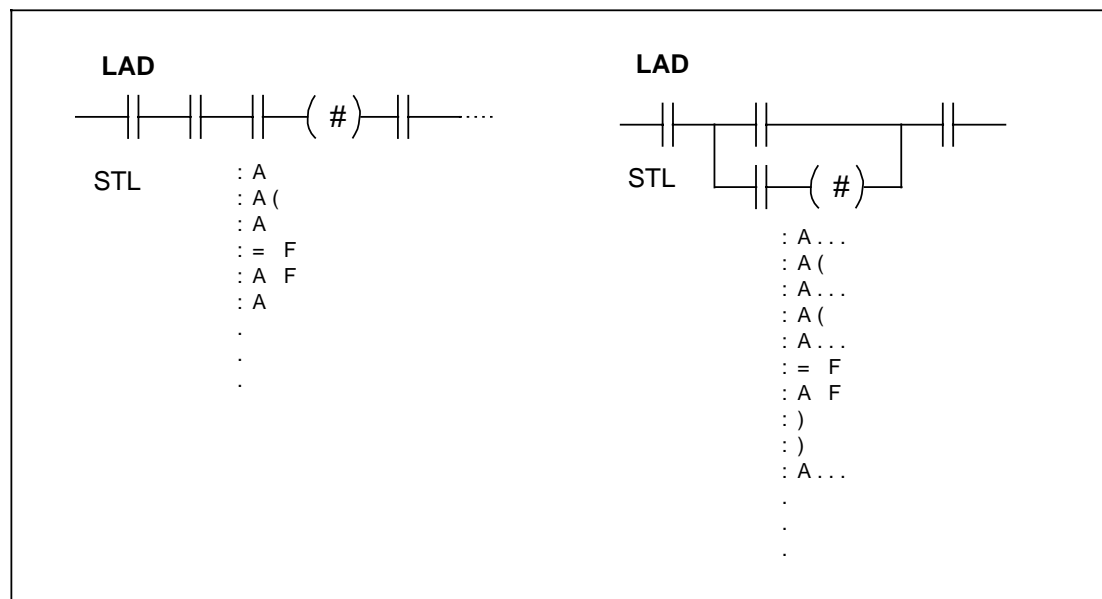


The connector in LAD and STL

a) Connectors with LAD

A connector registers, as an intermediate store, the result of the logic operation from the operations programmed before it in its own circuit. The following rules apply:

- Connectors in series (in series with other connectors):  
In this case a connector is treated as a normal contact.
- Connector in a parallel branch:  
Within a parallel branch, a connector is treated as a normal contact. Additionally, the entire parallel branch must be enclosed in parentheses of Type O (. . .).
- A connector may never be located immediately following the circuit (connector as first contact) or immediately after the opening of a circuit (connector as first contact in a parallel branch).



Connector controller for LAD

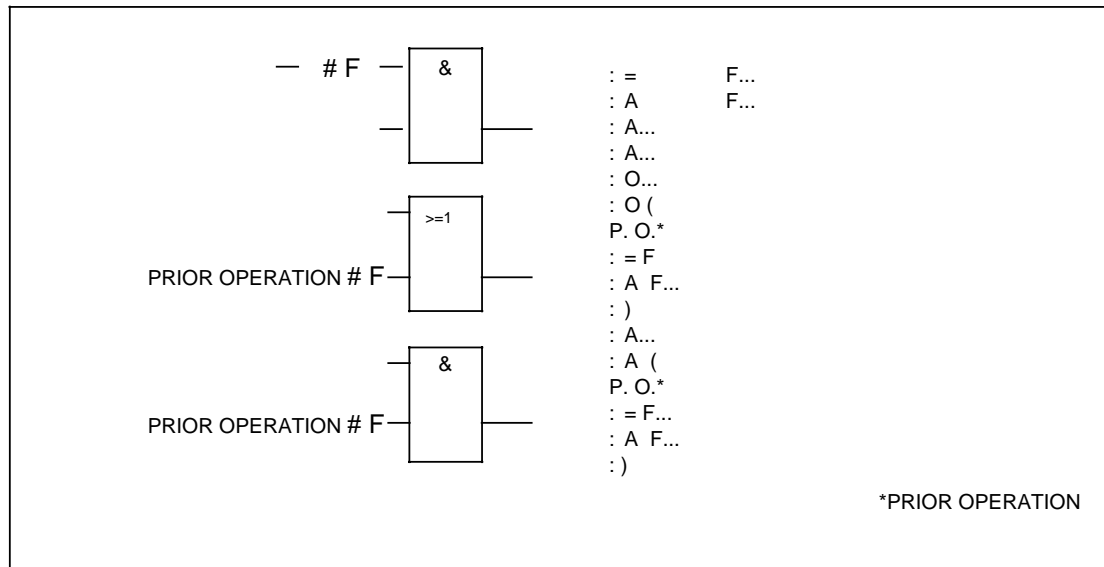
| CSF        | STL               |
|------------|-------------------|
| — # F... — | :=F...<br>:A F... |

The connector in CSF and STL

#### b) Connectors with CSF

A connector registers, as an intermediate store, the result of the entire binary operation preceding this connector. The following rules apply:

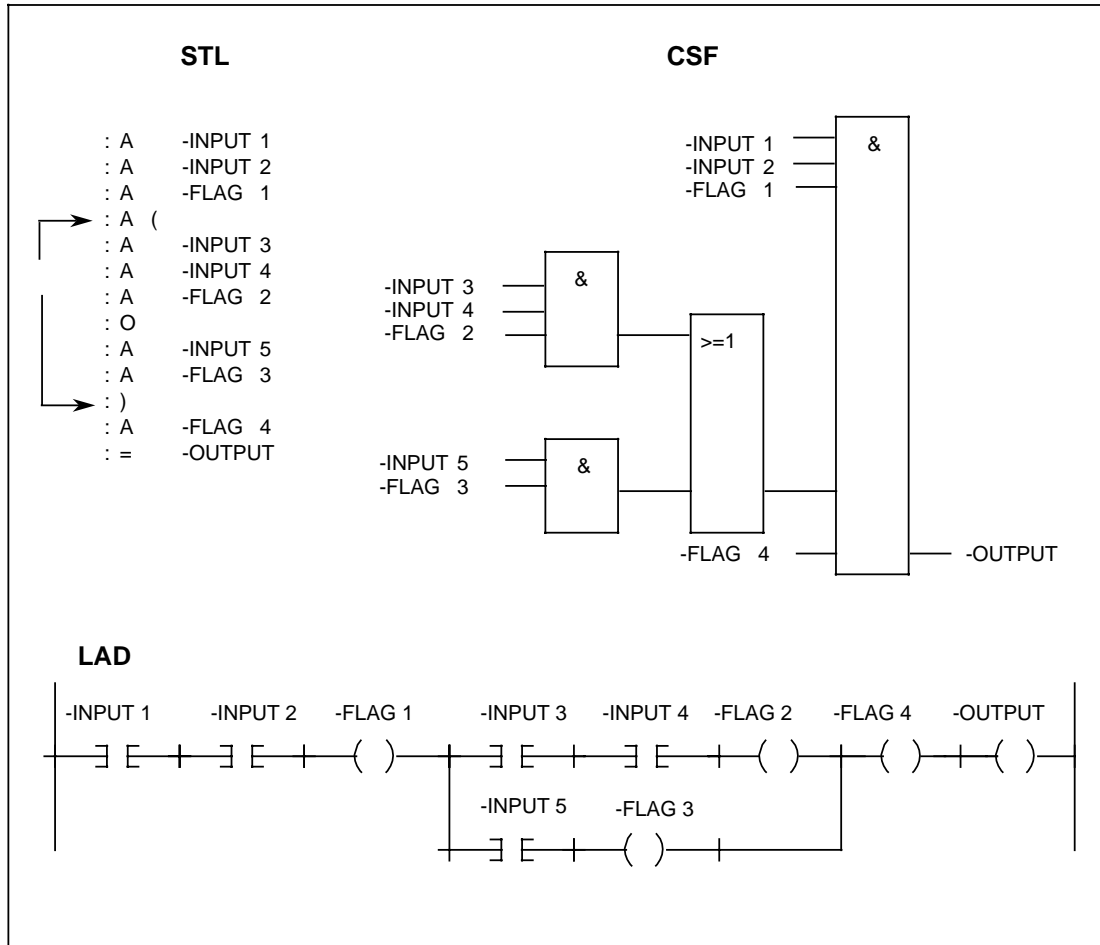
- Connector at the first input of an AND or OR-box:  
The connector is processed without parentheses.
- Connector not at the first input of an OR-box:  
The entire binary operation preceding the input is enclosed in parentheses of Type O (. . .).
- Connector not at the first input of an AND-box:  
The entire binary operation preceding the input is enclosed in parentheses of Type A (. . .). (Only allowed with CSF; not graphically representable with LAD.)



Connector controller for CSF

**Examples for connectors:**

Two examples are given: One without and one with connectors.



Example without connectors

Siemens AG

AUT V250  
P. O. Box 48 48  
W-8500 Nuremberg 1  
Federal Republic of Germany

**Suggestions**

**Corrections**

For Publication/Manual:

SINUMERIK 805  
Software version 4  
PLC Programming  
Planning Guide

Manufacturer Documentation

Order No.: 6ZB5 410-0CM02-0AA3  
Edition: November 1991

**From:**

Name \_\_\_\_\_

Company/Dept. \_\_\_\_\_

Address \_\_\_\_\_

Telephone / \_\_\_\_\_

Should you come across any printing errors when reading this publication, please notify us on this sheet. Suggestions for improvement are also welcome.

**Suggestions and/or corrections**

Siemens AG  
Automation Group  
Automation Systems  
for Machine Tools, Robots  
and Special-Purpose Machines  
P. O Box 48 48, W-8500 Nuremberg 1  
Federal Republic of Germany

© Siemens AG 1990 All Rights Reserved  
Subject to change without prior notice

Siemens Aktiengesellschaft

Order No.: 6ZB5 410-0CM02-0AA3  
Printed in the Federal Republic of Germany  
251/222049 PJ 03920.

