Typ3 osa

ICL700
System description

Edition 103

Typ3 osa

ICL700
System description

1070 073 737-103 (98.12) GB

Reg. no. 16149-03

Contents

Safety instructions

Please read this manual before commissioning the ICL700. Store this manual in a place to which all users have access at any time.

Proper use

This manual contains all information required for the proper use of the control unit. For reasons of clarity, however, it cannot contain each and every detail about each and all combinations of functions. Likewise, as the control unit is usually part of a larger installation or system, it is impossible to consider each and any aspect of integration or operation.

The Typ3 osa is used to

- activate feed drives, spindles and auxiliary axes of a machine tool via SERCOS interface for the purpose of guiding a processing tool along a programmed path to process a workpiece (CNC). Furthermore, a PLC is required with appropriate I/O components which – in communication with the actual CNC – controls the machine processing cycles holistically and acts as a technical safety monitor.
- program contours and the processing technology (path feedrate, spindle speed, tool change) of a workpiece.

Any other application is deemed improper use!

The products described

- have been developed, manufactured, tested and documented in compliance with the safety standards. These products pose no danger to persons or property if they are used in accordance with the handling stipulations and safety notes prescribed for their configuration, mounting, and proper operation.
- comply with the requirements of
  - the EMC Directives (89/336/EEC, 93/68/EEC and 93/44/EEC)
  - the Low-Voltage Directive (73/23/EEC)
  - the harmonized standards EN 50081-2 and EN 50082-2
- are designed for operation in industrial environments. For operation in residential environments, in trade and commercial applications and small enterprises, an individual permit of the national authority or test institution is required; in Germany, please contact the Bundesanstalt für Post und Telekommunikation or its local branch offices.

The faultless, safe functioning of the product requires proper transport, storage, erection and installation as well as careful operation.

Qualified personnel

The requirements as to qualified personnel depend on the qualification profiles described by ZVEI (central association of the electrical industry) and VDMA (association of German machine and plant builders) in:
Weiterbildung in der Automatisierungstechnik
edited by: ZVEI and VDMA
MaschinenbauVerlag
Postfach 71 08 64
D-60498 Frankfurt

The present manual is designed for NC project engineers and PLC programmers. They need special knowledge on performance data, system behavior and available PLC instructions.

Programming, start and operation as well as the modification of program parameters is reserved to properly trained personnel! This personnel must be able to judge potential hazards arising from programming, program changes and in general from the mechanical, electrical, or electronic equipment.

Interventions in the hardware and software of our products, unless described otherwise in this manual, are reserved to our specialized personnel.

Tampering with the hardware or software, ignoring warning signs attached to the components, or non-compliance with the warning notes given in this manual can result in serious bodily injury or property damage.

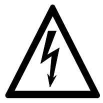
Only electrotechnicians as recognized under VDE 1000-10 who are familiar with the contents of this manual may install and service the products described. Furthermore, all existing accident prevention regulations (in Germany: UVV VBG4, VDE 100, VDE 105) and installation instructions (EN 60204-Part 1, EN 50178) must be observed.

Such personnel are

- those who, being well trained and experienced in their field and familiar with the relevant norms, are able to analyze the jobs being carried out and recognize any hazards which may have arisen.
- those who have acquired the same amount of expert knowledge through years of experience that would normally be acquired through formal technical training.

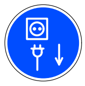
Please note our comprehensive range of training courses.
Our training center will be pleased to provide you with further information, telephone: ++49 (0) 6062 78-258.

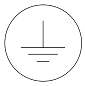
Safety markings on products


Warning of dangerous electrical voltage!


Warning of danger caused by batteries!


Components sensitive to electrostatic discharge!


Disconnect from mains before opening!


Pin for connecting PE conductor only!


Connection of shield conductor only

Safety instructions in this manual

DANGEROUS ELECTRICAL VOLTAGE
This symbol is used to warn of a **dangerous electrical voltage**. The failure to observe the instructions in this manual in whole or in part may result in **personal injuries**.

DANGER
This symbol will be used if the failure to observe the instructions in this manual in whole or in part may result in **personal injuries**.

CAUTION
This symbol will be used if the failure to observe the instructions in this manual in whole or in part may result in **damages to equipment or data files**.

☞ This symbol will be used to draw the user's attention to special circumstances.

★ This asterisk symbol refers to an activity to be performed by the user.

Key operation

Special keys or combinations of keys are represented by pointed brackets

- Special keys: e.g. <enter>, <pgup> , <del>
- Key combinations (pressed simultaneously): e.g. <ctrl>+<pgup>

Changes

Modifications in the present operating instructions as compared to issue 102 are marked by black vertical bars on the margin.

Safety instructions

⚠

DANGER
Danger for persons and equipment!
First every new program should be thoroughly tested without axis movement! In the "Automatic" group operating mode, the control offers the opportunity to inhibit axis movements and/or the output of auxiliary functions by pressing certain softkeys.

⚠

DANGER
Danger of life through inadequate EMERGENCY-STOP devices!
EMERGENCY-STOP devices must be active and within reach in all system modes. Releasing an EMERGENCY-STOP device must not result in an uncontrolled restart of the system! First check the EMERGENCY-STOP circuit, then switch the system on!

⚠

DANGER
Retrofits or modifications may adversely affect the safety of the products described!
The consequences may include severe injuries, damage to equipment, or environmental hazards. Possible retrofits or modifications to the system using third-party equipment therefore have to be approved by Bosch.

DANGER
Movements of tools or axes may cause serious or fatal injuries!

Feed and spindle motors generate very powerful mechanical forces and can accelerate very quickly due to their high dynamics. You should therefore always stay outside the danger area of the machine when it is running!

Do not ever – not even briefly – deactivate the safety-relevant functions of the unit!

Report any malfunction of the unit to your servicing and repairs department immediately.

Inappropriate working clothes may cause serious or fatal injuries! During careless handling of machines with rotating parts, clothes or long hair may get caught in the mechanics, pulling operators into the machine! Therefore:

- **Wear a hair net!**
- **Wear a protective suit!**
- **Take off protective gloves before working near moving parts!**
- **Take off any jewelry and wristwatches!**

Remember that chippings, borings, etc. may be cast out during operation of the machine! They can cause eye injuries and burns. Therefore:

- **Wear protective goggles!**
- **Wear a protective suit !**

There is also a risk of injury from sharp edges on the workpieces and tools! Therefore:

- **Wear protective gloves!**

DANGEROUS ELECTRICAL VOLTAGE
Unless described otherwise, maintenance works must be performed on inactive systems! The system must be protected against unauthorized or accidental reclosing.

For measuring or test activities on the live system, the existing safety and accident prevention regulations must be observed in any case. Use suitable insulated tools for all types of work!

CAUTION
Repair/maintenance work is reserved to the Bosch service or repair/ maintenance units authorized by Bosch!
Only replacement/spare parts approved by Bosch may be used!

ACHTUNG
Beim Umgang mit Baugruppen und Bauelementen alle Vorkehrungen zum ESD-Schutz einhalten! Elektrostatische Entladungen vermeiden!

The following protective measures must be observed for modules and components sensitive to electrostatic discharge (ESD)!

- The personnel resposible for storage, transport, and handling must have been trained for ESD protection.
- ESD-sensitive components must be stored and transported in their prescribed protective packaging.
- ESD-sensitive components may only be handled at special ESD-workplaces.
- Personnel, working surfaces, as well as all equipment and tools which get in contact with ESD-sensitive components must have the same potential (e.g., by grounding).
- Wear an approved grounding bracelet. The grounding bracelet must be connected with the working surface through a cable with an integrated resistor of 1 MΩ.
- ESD-sensitive components must by no means get in contact with chargeable objects, including most plastic materials.
- When ESD-sensitive components are installed in or removed from equipment, the equipment must be de-energized.

Documentation and Software

The present manual provides information on the general performance data, the system behavior and the instruction set of the ICL700.
The following comprehensive documentation is available for this series of controls:

Typ3 osa documentation	Part no. German	English
Interface conditions for project engineering and maintenance	1070 073 704	1070 073 736
Operating instructions – Standard operator interface	1070 073 726	1070 073 739
Operating instructions – Diagnostics	1070 073 779	1070 073 780
DIN programming instructions for programming to DIN 66025	1070 073 725	1070 073 738
CPL programming instructions	1070 073 727	1070 073 740
ICL700 system description, program structure of the integrated PLC	1070 073 706	1070 073 737
ICL700 project engineering manual, software interfaces and CNC interface signals of the integrated PLC	1070 073 728	1070 073 741
MACODA operation and configuration of the machine parameters	1070 073 705	1070 073 742

☞ **The current release number of the individual software modules can be viewed by selecting the ”Control-Diagnostics” softkey in the ”Diagnostics” group operating mode.**

☞ **The software version of Windows95 or WindowsNT is displayed by selecting the ”My Computer” icon on the Start screen or in the HELP menu, item ”About Windows95” or ”About WindowsNT”.**

1 General performance data

Processors

- 80386 DX processor for word processing and high-level language
- bit logic processor for fast bit processing 0.24 ... 0.5 ms/k instruction
- 80186 processor for interface processing

Application memory

- 512 kbyte

Program execution

- cyclic
- interrupt controlled
- time-controlled

Programming

- instruction list
- Ladder diagram
- Sequential function chart
- High-level language C
  – universal symbol definition
  – universal module concept

Operands

- 2k inputs
- 2k outputs
- 48k markers
- 256 timers
- 256 counters
- 512 byte data buffer, also bit-addressable
- 32k byte data field, also bit-addressable
- 512 data modules, also bit-addressable, 2 of which simultaneously active
- 1k byte user stack, 256 DW
- 128k byte communication memory

Communication

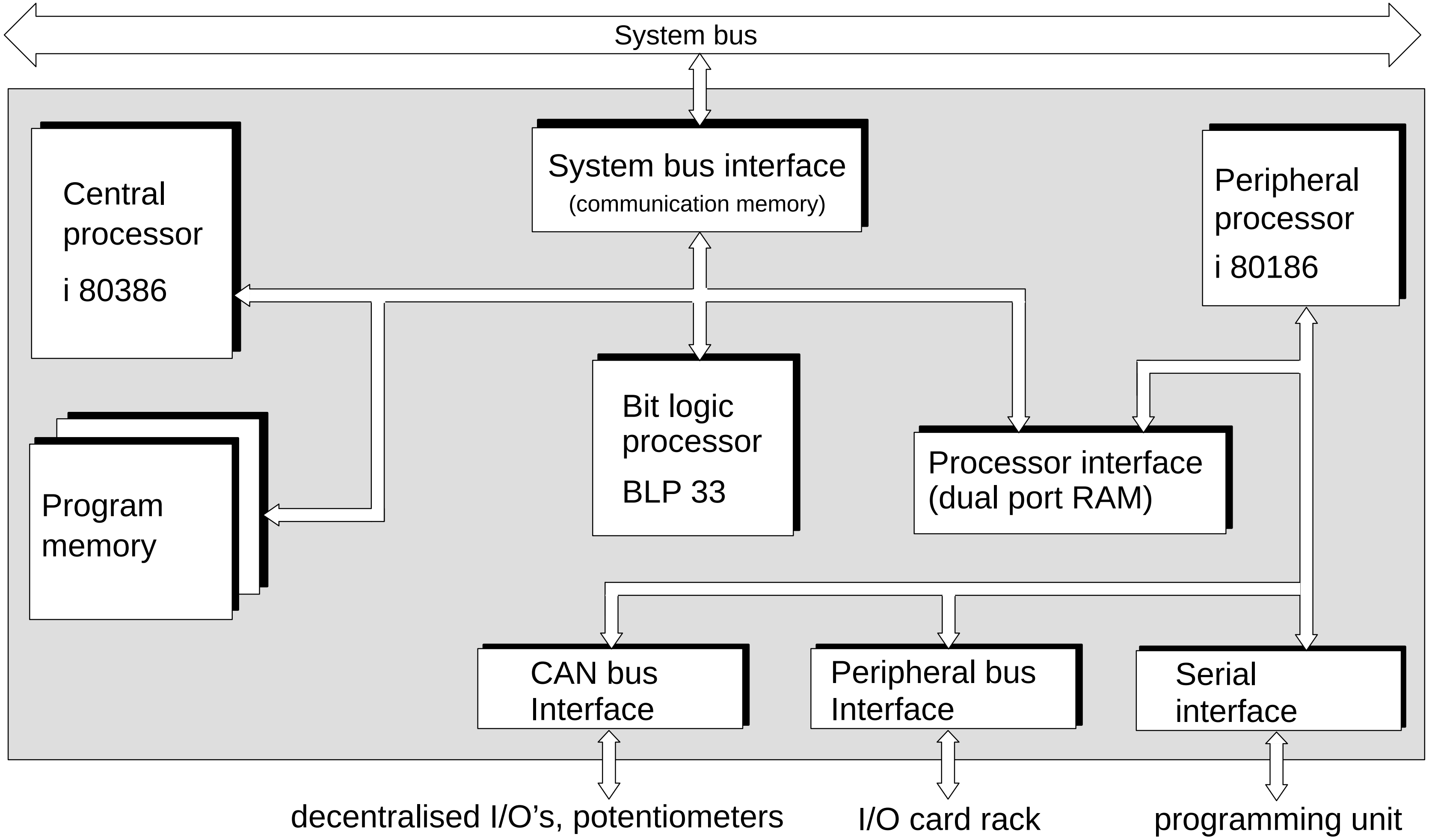
- with NC
  – via communication memory (DPR)
- with peripherals
  – via interface to I/O card rack
  – via CAN bus (decentralised I/Os, potentiometers)
  – via serial interface to PG

Monitor

- dynamic monitor function (on line)
- static monitor function (single step)
- debug function for C program modules

High-level language

- Language C
- use of standard compilers,
  e.g. Metaware C version 3.x, GNU
- full integration in PLC programming interface
- operation of C modules in established module design
- library modules for PLC-specific functions
- high-level language monitor similar to well-known high-level language debuggers, e.g. Codeview
- flexibly adjustable real-time performance in monitor operation


System bus
System bus interface
(communication memory)
Central processor
i 80386
Peripheral processor
i 80186
Bit logic processor
BLP 33
Program memory
Processor interface
(dual port RAM)
CAN bus Interface
Peripheral bus Interface
Serial interface
decentralised I/O’s, potentiometers
I/O card rack
programming unit

Your notes:

2 System performance

2.1 Program structure

The modular design of control unit ICL700 enables the user to divide programs into functionally related blocks, i.e. to organise programs in a structured manner. Various types of modules are available for this purpose.

2.1.1 Module types

ICL700 has the following module types:

- organisation modules
- program modules (parameterisable)
- data modules

Organisation modules (OM)

Organisation modules establish the connection between the system program and the user program. Organisation modules are programmed like program modules but may be called only by the system program.

Modules can be called depending on certain conditions. They may be called only when they need to be processed.

The user also has access to organisation modules in which he may specify the response to particular events, such as interrupts, operating time or start-up performance.

Organisation modules are not parameterisable.

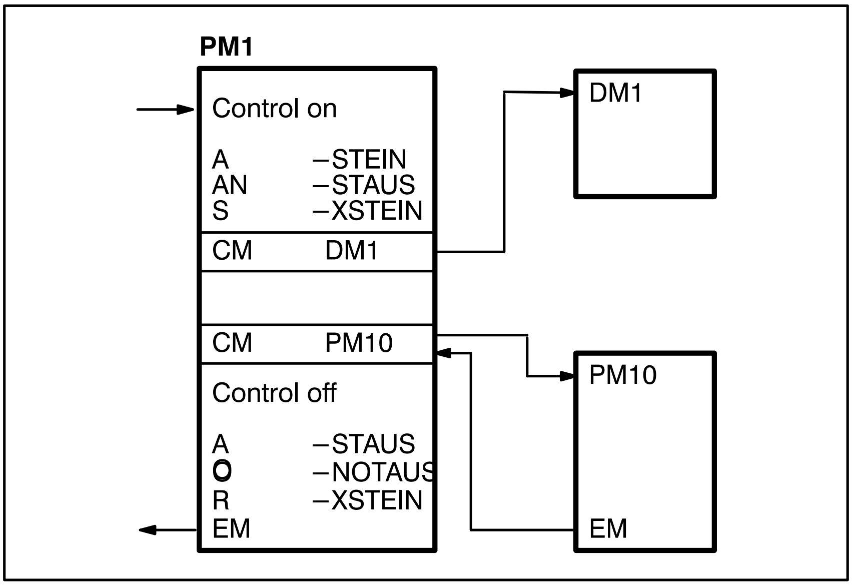
The following organisation modules are available:

- OM for initialisation table (OM2)
- OM for cyclic program execution (OM1)
- OMs for cold start and restart (OM5, OM7)
- OMs for time-controlled program execution (OM18-21)
- OMs for peripheral interrupt-controlled program execution (OM10-13)
- 16 OMs for library modules (OM48-63)
- 1 OM for error management (OM9)

Program modules (PM)

Program modules contain technologically or functionally related program parts, e.g. program modules for tool changers, feeding device, etc. Program modules are called by organisation modules or other program modules.

Program modules are parameterisable. Up to 63 current parameters can be transferred together with a module call.
Program modules can be realised in IL, sequential function chart, ladder diagram or in programming language “C”.



Examples of module calls in PM1

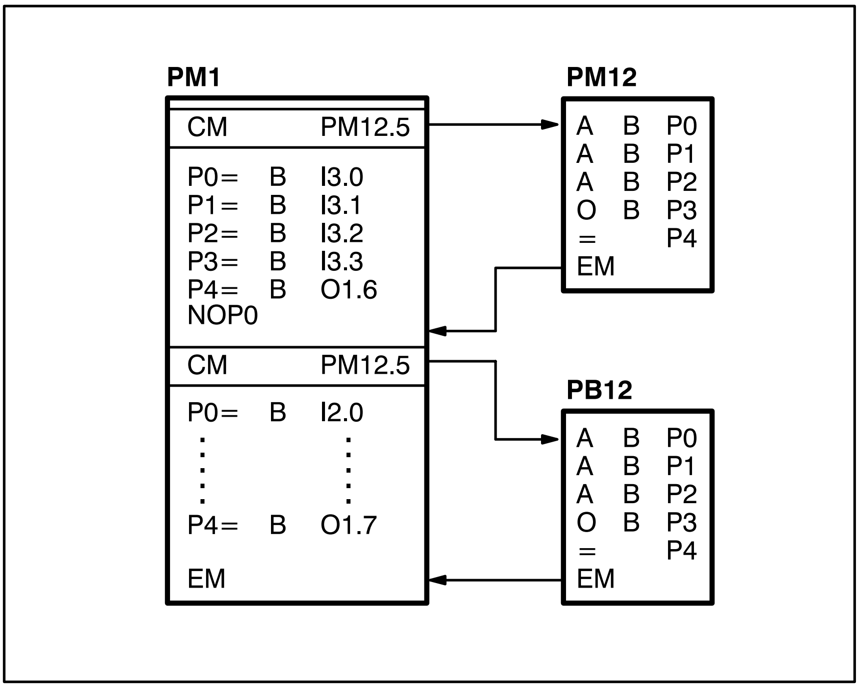
The following parameters are possible:

- Input parameters: I, O, M, SM, II, EI, T, C, K, S various formats, D, DX, DB, DF, PM, DM (in absolute and symbolic addressing)
- Output parameters: O, M, IO, EO, IO, DF, DB, S, D, DX, T, C

A total of 1024 program modules are available.

For program modules, a nesting depth of 64 with organisation module OM1 is permitted.

Program modules are usually concluded with EM (end of module). If EP (end of program) is selected, the program is aborted and an input/output cycle is initiated.



Example of program module programming

Data modules (DM)

All fixed and variable data can be stored in the data modules by the user.

Using the initialisation table (OM2), the user can copy any desired data module from the user program memory into a data buffer area DB. Data modules can be written to and read by the PLC program.

Data field (DF)

The data field is stored in a buffered RAM area and is also available as a variable data field.

The data field is made up of 32 Kbytes.

512 data modules are available, two of which may be simultaneously active.

Example:	1st DM	Call:	CM		DM1
		Access:	L	W	D0,A
	2nd DM	Call:	CX		DM2
		Access:	L	W	DX0,A

DM 1 Name:AST2DB1 Comment: DB1F.ABLST3 MAIN BRANCH RAM/EPROM: R

No.		Symbol	Type	Sign	Data field	F
D	24	SCHR6HP	Word	N	6	H
D	26		Word	N		B
D	28		Word	N	0000000000000000	B
D	30		Word	N	0000000000000000	B
D	32		Word	N	0000000000000000	B
D	34		Word	N	01010101010000	B
D	36		Word	N	0000000000000000	D
D	38		Word	N	12345	D
D	40	DBNRHP	Word	N	0001	H
D	42	SCHANZHP	Word	N	0600	H
D	44	SCHVBIHP	Word	N	0000000000000000	H
D	46	SANZHP	Word	N	0	H
D	48	SANZ1HP	Word	N	0	H
D	50	BAMLDGHP	Word	N	0	B

Example of data module programming

Data buffer

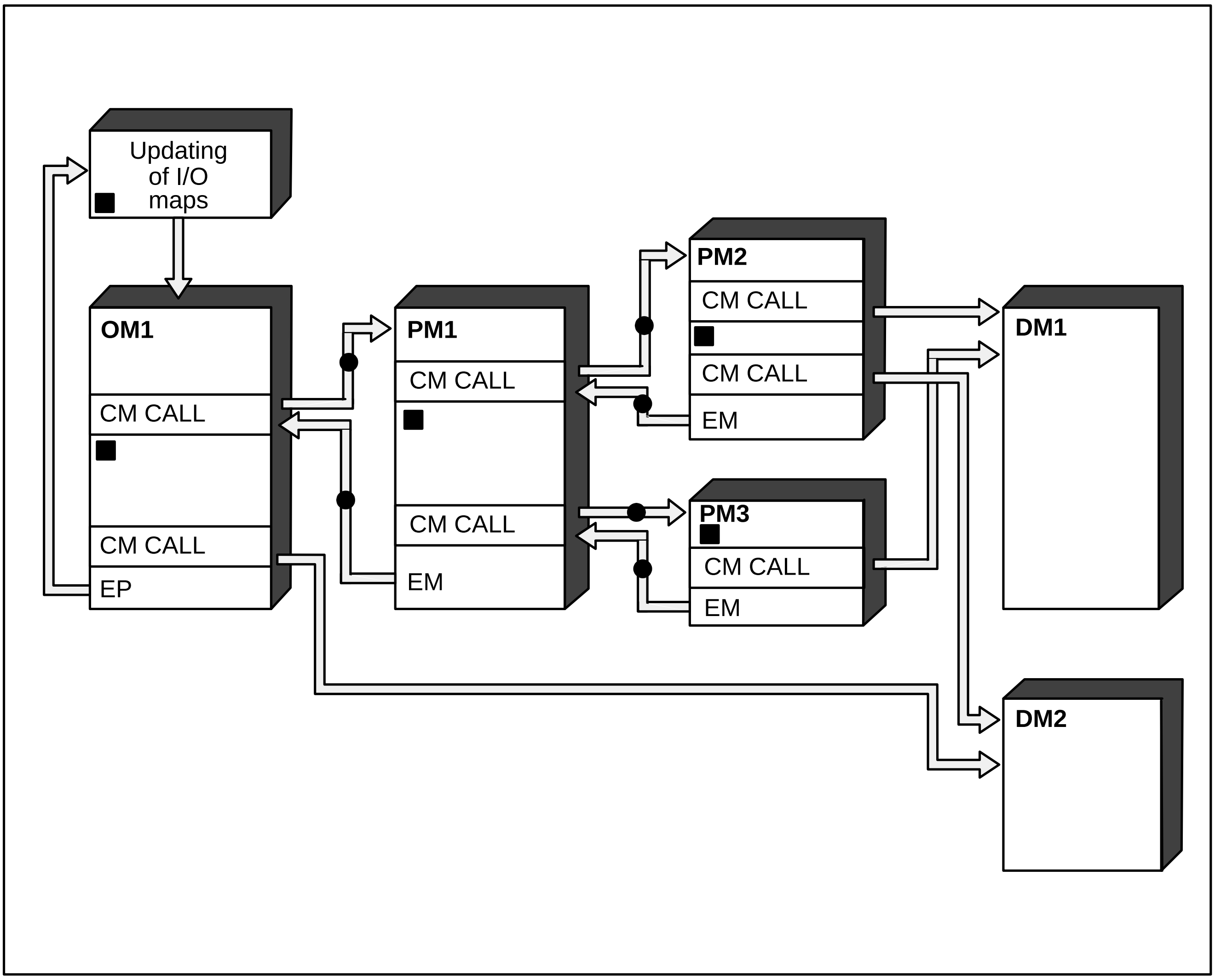
The data buffer is in a buffered RAM area and is also available as a variable data buffer.

The data buffer is made up of 512 bytes.

Summary of module types

	OM	PM	DM
Number of modules	64	1024	512
Parameterisable	–	up to 63 parameters	–
Nesting depth	64 modules incl. OM1		

Example of structured relationship of module types

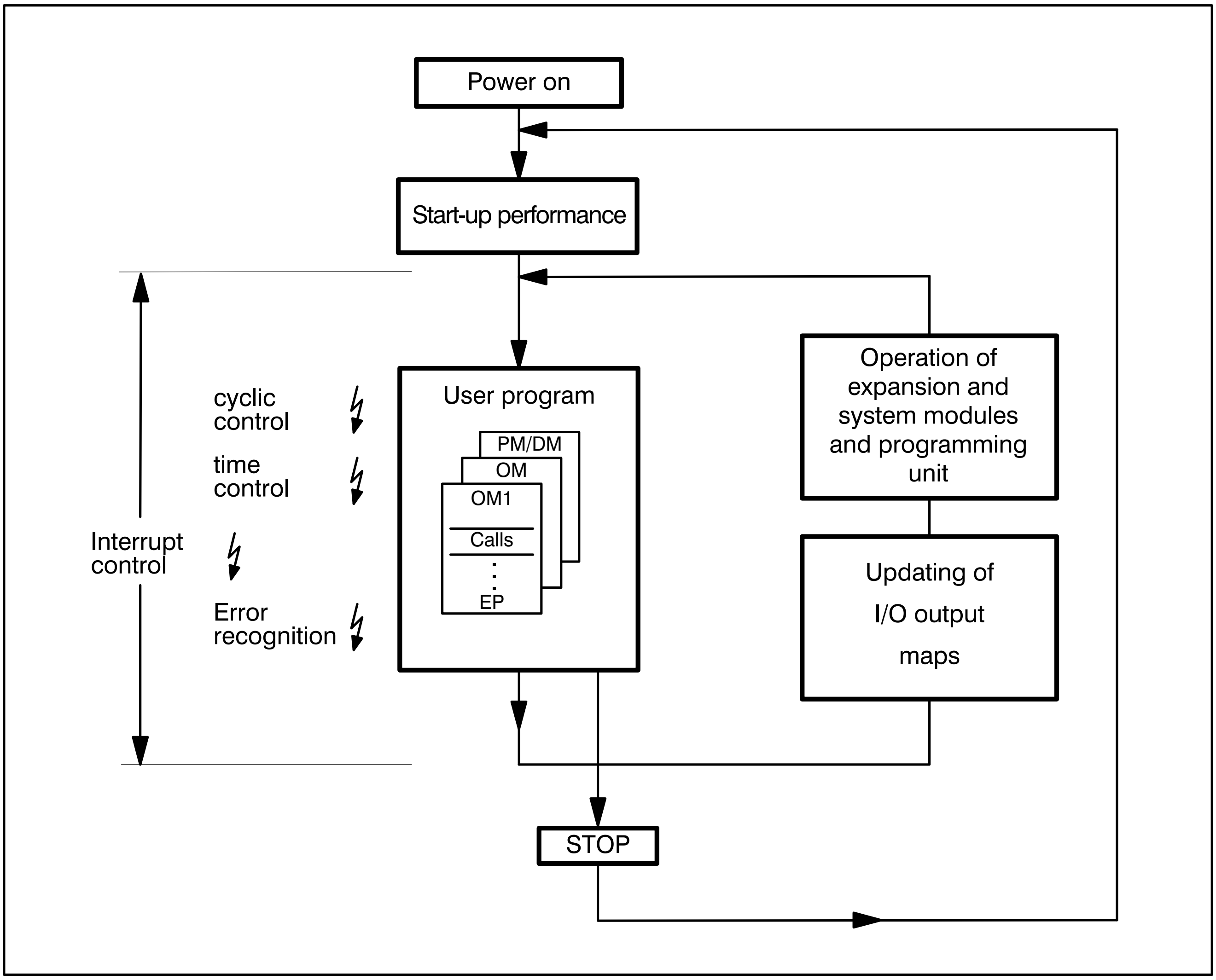


Example of structured relationship of module types

Program interruption points where a time-controlled processing can be inserted are identified with "●".

"■" indicates program interruption points where an interrupt can be inserted.

2.2 Program execution



ICL700 flow diagram

2.3 Module start-up

The ICL 700 module is booted after **power “on”** or by **module stop** if Typ 3 is in the relevant operating condition. The position of the rotary switch on the SMNC determines the operating condition and consequently whether or not the ICL 700 is booted.
Please ensure that booting the ICL 700 does **not** correspond to the start-up performance.
There are two program parts: 1. System program
2. User program
The system program is always loaded to the ICL and started by the SMNC. The user program can be loaded to the ICL and started either by the programming unit or by the SMNC. An automatic start-up without reloading the user program is also possible.

The rotary switch on the SMNC determines the booting behaviour of the ICL 700:

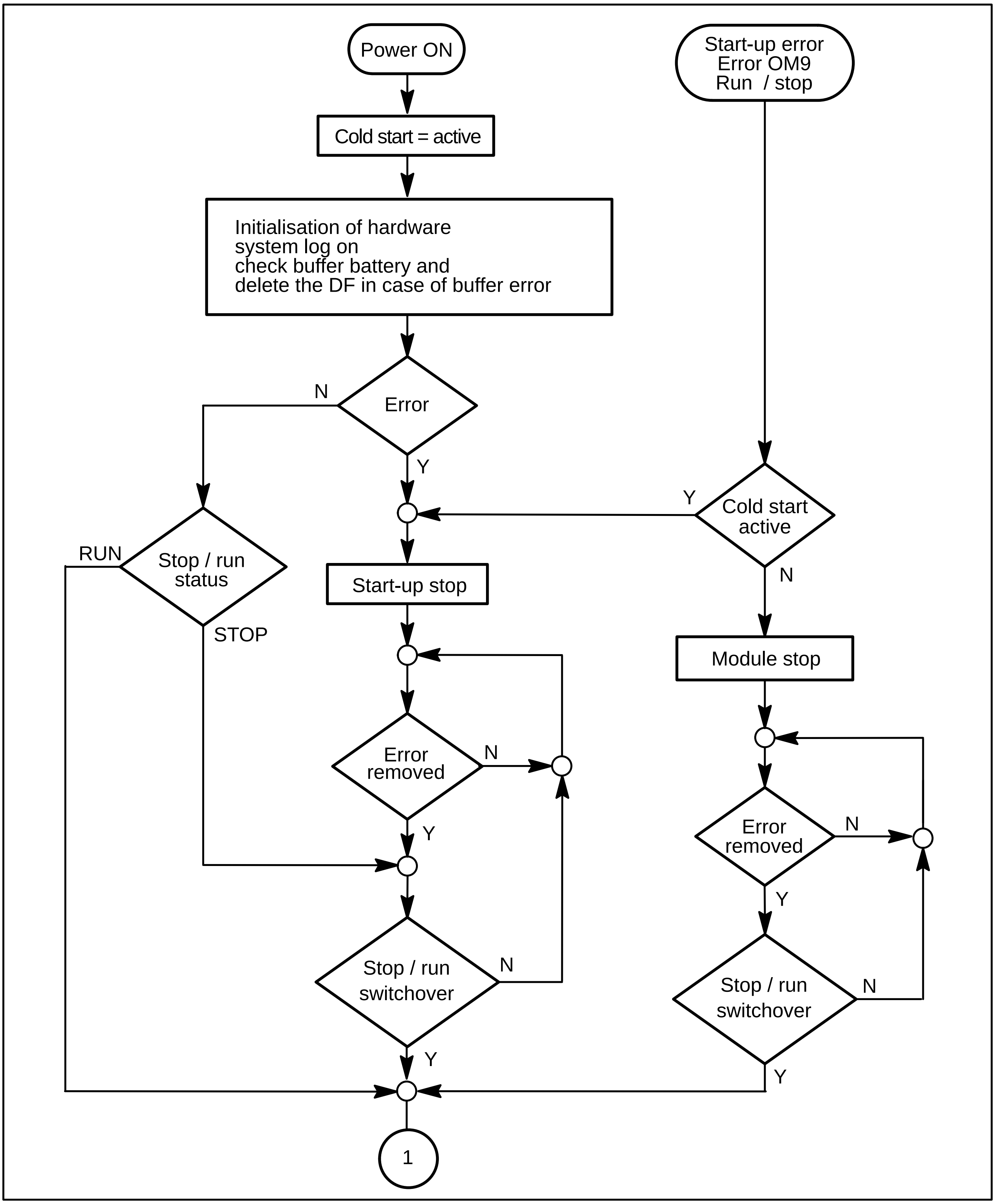
Switch setting	Mode	Explanation
0	Normal operation RUN	System firmware is loaded by the SMNC, user program of ICL 700 is started
1	Normal operation STOP	System firmware is loaded by the SMNC, user program of ICL 700 is **not** started
6	Boot-strap	System firmware is loaded by the SMNC, user program is loaded and started from the SMNC (if present) or must be loaded via the programming unit or the user interface computer
...C	System security	System firmware can be saved by the programming unit to the SMNC

If the ICL 700 is in **stop mode,** it will start up only if

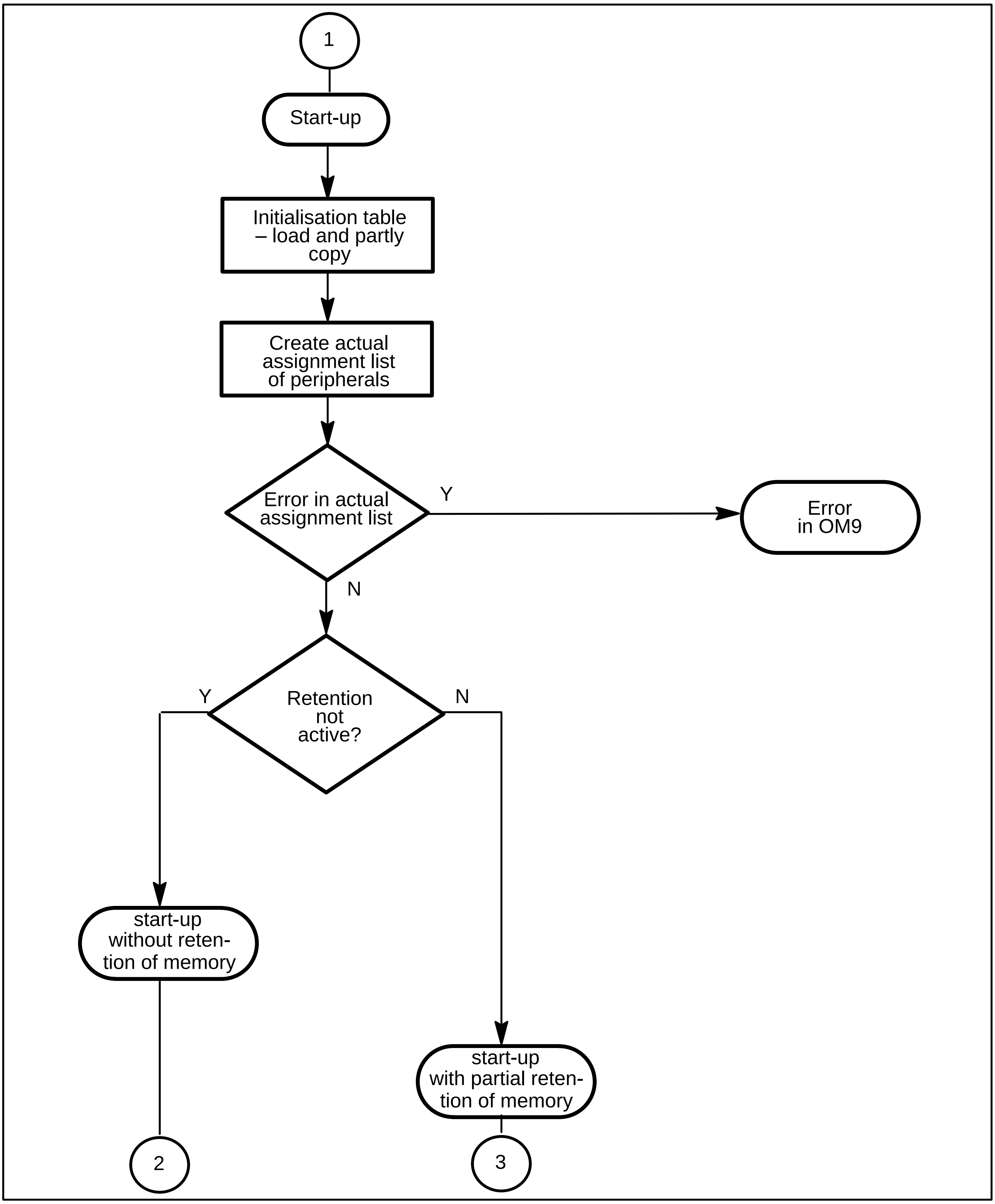
- the **cause of the stop** has been removed and it is switched to **run mode**
- or it is switched to **run mode** by means of the software.

The following diagram shows the cold start and restart of the ICL 700

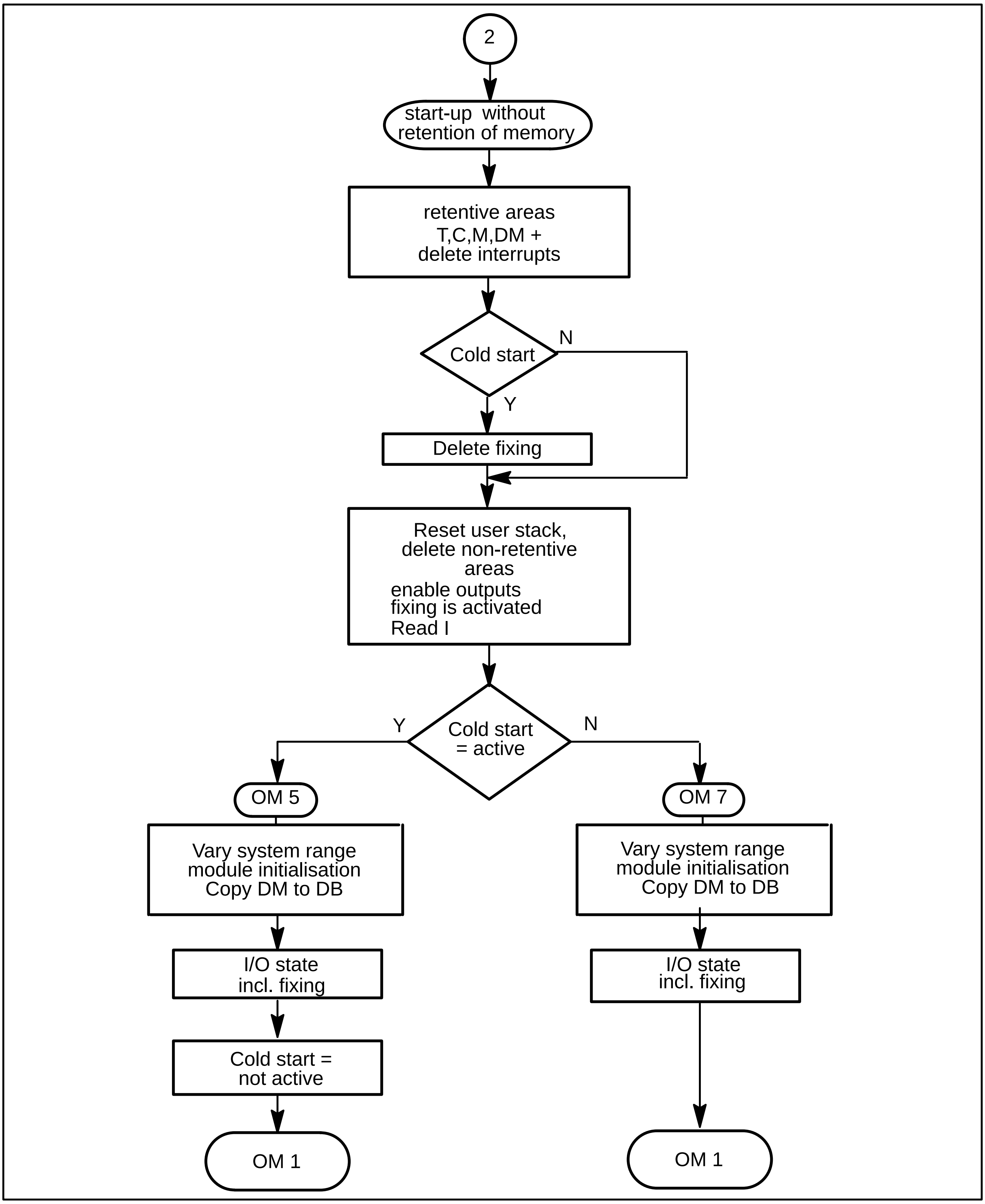
- without retention of memory and
- with partial retention of memory.



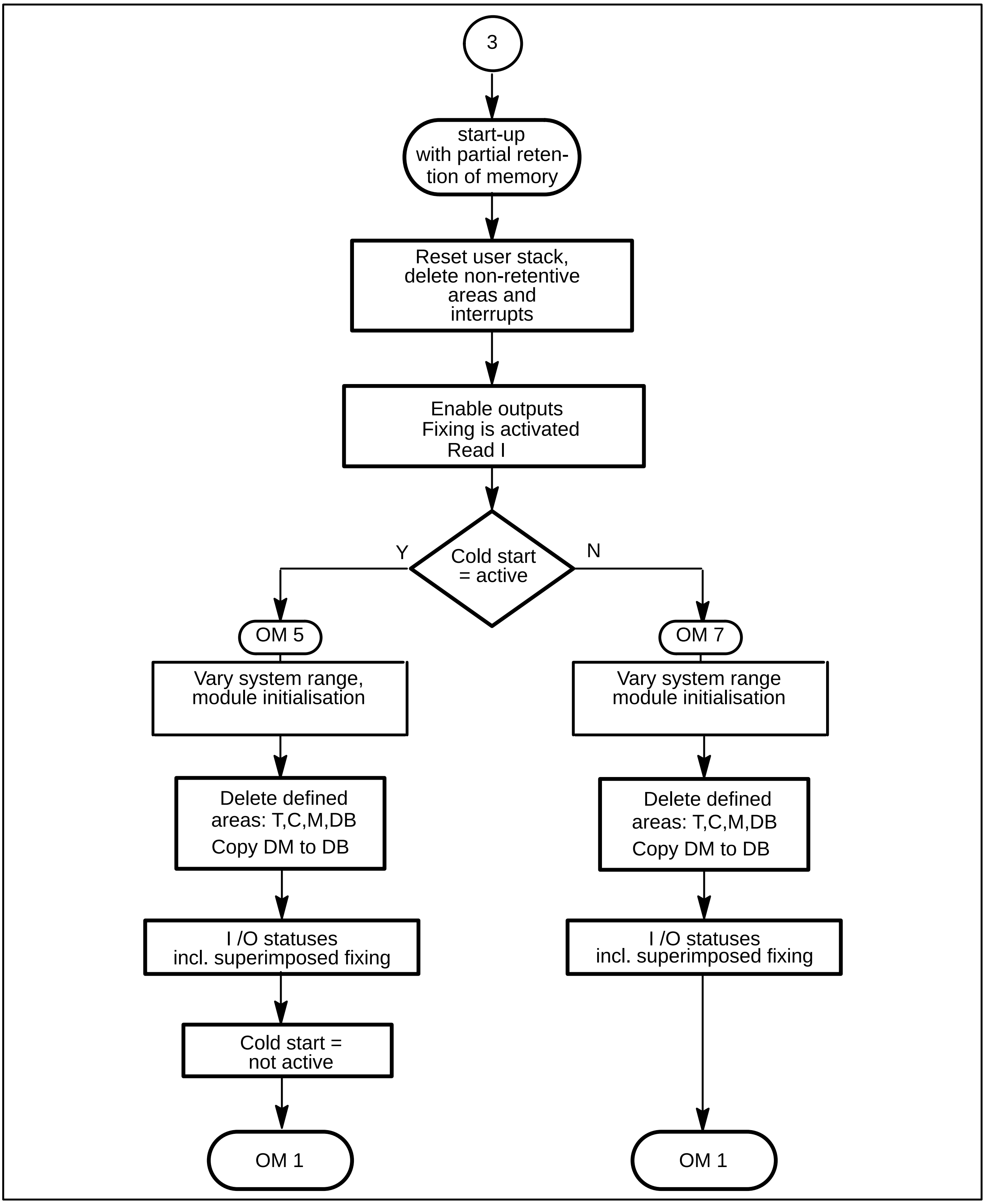
Module start-up – sheet 1



Module start-up – sheet 2



Module start-up – sheet 3



Module start-up – sheet 4

2.4 Start-up via OM5 and OM7

OM5 is executed after **power on,** insofar as it is integrated in the program.

OM7 is executed when the mains power supply is permanently switched on, if there is a software switch from **stop** to **run** and OM7 is integrated in the program.

The data declared as retentive are retained or deleted according to the retention setting in OM2.

- **Setting non-retentive**
  - All marker, counter, timer and data buffer inputs are deleted **before** start-up.
- **Setting partially retentive**
  - Data declared as non-retentive is deleted **after** start-up. Data declared as retentive is retained.

☞ **No non-retentive operands can be preset in partially retentive operation in the OM start-up.**

2.5 Definitions in OM2

OM2 contains the initialisation table in which the values of the ICL700 firmware can be varied. It is edited by the user using the program editor.

OM2 must be integrated in every ICL700 user program which uses non-standard settings and then stored in the user memory. If there is no OM2 entry in the symbol file, the standard settings are used.

Parts of OM2 are copied to the system range and can be manipulated by the user. The final module initialisation is performed with the values from OM2 and the system range before program execution is begun.

The assignment of the system range is portrayed in the ICL700 operation list.

☞ **Any amendment of data words in prohibited address ranges can result in undefined system performance of the ICL 700.**

The declarations made in OM2 are copied to the system range during start-up. Manipulations can be performed here by the PLC program, e.g. the time reference for time-controlled OMs can be changed by an appropriate entry. The complete assignment of the system range is portrayed in the "Operation list" chapter.

Data areas in OM2

Data word 1

reserved

DEFW K0000000000000000B

Data word 2

Data word 2 is intended for the initialisation flags.

Entry 0 = Do not check or perform function
Entry 1 = Check or perform function

DEFW K0000000000000000B

The bits of data word 2 are divided as follows:

Bit 0: Check assignment list

This determines whether the actual assignment list is to be compared with the created desired assignment list. If an error is detected during comparison, a branch to error module OM9 takes place (if present) or the module returns to STOP status.

Bit 1: Check cycle time

The cycle time is monitored by the ICL 700 for a permanently set maximum value of 1.6 s (set by hardware). Below this value, a further cycle time limit can be specified in OM2. If the cycle time is exceeded, the special marker bit SM29.7 is set and OM9 is called, where the user can program the response to cycle time monitoring. When OM9 has been processed, or if it has not been programmed, the ICL 700 goes into STOP status and all outputs are reset.

Bit 2: **start-up with partial retention of memory**

Bits 3 to 6 are not assigned

Bit 7: **outputs are output during start-up with partial retention of memory**

Here it is determined that during start-up at the interruption position the outputs are to be output immediately and not in the next I/O state

Bit 8: **copy data module to data buffer**

Here the user can specify whether he wishes a particular data module to be copied from the user program to the data buffer during ICL700 start-up.

Bits 9 to 15 are not assigned.

Data word 4 (currently unused)

Data word 4 is intended for the response to errors.
Here it is determined whether an error should result in Typ 3 system stop or only in module stop of the ICL.

Entry 0 = Do not perform system stop in case of error
Entry 1 = Perform system stop in case of error

DEFW K0000000000000000B

Bits 0 to 7 are not assigned

Bit 8: **partial retention of memory during start-up desired, but forbidden**

Bit 9: **conflict in assignment list**

Bit 10: **direct access takes too long**

Bit 11: **program error**

Bit 12: **HOLD command**

Bit 13: **stop request from PG or from system manager**

Bit 14: **stop switch**

Bit 15: **not assigned**

Data word 5

Data word 5 is intended for cycle time monitoring (time value x time reference 10 ms).

Entries from **K1D** to **K200D** (10 to 2000 ms) with regard to cycle time monitoring are possible.

The function is performed if **bit 1 = 1** in data word 2.
DEFW K200D = max. cycle time

Data word 6

The number of the data module to be copied to the data buffer is entered in data word 6.

Possible entries 0 to 512.
The function is performed if **bit 8 = 1** in data word 2.

DEFW K0D = copy DM0 to data buffer

Data word 7

Data word 7 is intended for the first retentive timer. Entries from K0D to K256D possible (K64D = retention for the timer loops T64 to T255)

K256D = no retention

DEFW K64D = first retentive timer T64

Data word 8

Data word 8 is intended for the first retentive counter. Entries from K0D to K256D possible (K64D = retention for the counters C64 to C255)

K256D = no retention

DEFW K64D = first retentive counter C64

Data word 9

Data word 9 is intended for the first retentive marker. Entries from K0D to K6144D possible (K128D = retention from marker byte M128/marker bit M128.0. The retention limit is defined via byte addresses)

K6144D = no retention

DEFW K128D = first retentive marker byte 128 = M128.0

Data word 10

Data word 10 is intended for the first retentive address in the data buffer. Entries from K0D to K512D possible (K256D = retention from data buffer byte DB256)

K512D = no retention

DEFW K256D = first retentive address in data buffer DB256

Data words 11 to 14

Data words 11 to 14 are intended for the definition of time (time value x time reference 10 ms) in the OMs 18, 19, 20, 21 for time-controlled processing. Entries from K1D to K65535D are possible.

K0D = no time-controlled processing.

K1D = 1 x 10 ms = 10 ms time interval for time-controlled processing

Data word 11 for OM18

DEFW K0D

Data word 12 for OM19

DEFW K0D

Data word 13 for OM20

DEFW K0D

Data word 14 for OM21

DEFW K0D

Data words 15 to 18

Data words 15 to 18 are intended for the system memory. The system memory may not be changed by the user.

Data word 15 (standard value K0000H)

DEFW K0000H

Data word 16 (standard value K0000H)

DEFW K0000H

Data word 17 (standard value K0000H)

DEFW K0000H

Data word 18 (standard value K0000H)

DEFW K0000H

Data words 19 to 23

reserved

Data words 24 to 32

reserved

Data words 33 to 48

Data words 33 to 48 are intended for the definition of peripheral input assignment lists.

Every input byte assigned in the control unit is identified by a “1” in the data word.

A “0” indicates that the input byte has not been assigned. 16 input bytes per data word are identified as assigned or not assigned.

Data word 33 for assignment list of the input byte 15 ... 0

DEFW K0000000000000000B

Example: K0000000000000001B = I byte 0/I0.0 to I0.7 present

Data word 34 for assignment list of the input byte 31 ... 16

DEFW K0000000000000000B

Example: see data word 33

Data word 35 for assignment list of the input byte 47 ... 32

DEFW K0000000000000000B

Example: see data word 33

Data word 36 for assignment list of the input byte 63 ... 48

DEFW K0000000000000000B

Example: see data word 33
.
.

Data word 48 for assignment list of the input byte 255 ... 240

DEFW K0000000000000000B

Example: see data word 33

Data words 49 to 64

Data words 49 to 64 are intended for the definition of peripheral output assignment lists.

Every output byte assigned in the control unit is identified by a “1” in the data word.

A “0” indicates that the output byte has not been assigned. 16 output bytes per data word are identified as assigned or not assigned.

Data word 49 for assignment list of the output byte 15 ... 0

DEFW K0000000000000000B

Example: K0000000000000001B = O byte 0/O0.0 to O0.7 present

Data word 50 for assignment list of the output byte 31 ... 16

DEFW K0000000000000000B

Example: see data word 49

Data word 51 for assignment list of the output byte 47 ... 32

DEFW K0000000000000000B

Example: see data word 49

Data word 52 for assignment list of the output byte 63 ... 48

DEFW K0000000000000000B

Example: see data word 49
.
.
.

Data word 64 for assignment list of the output byte 255 ... 240

DEFW K0000000000000000B

Example: see data word 49

Data words 65 to 80

Data words 65 to 80 are intended for the definition of peripheral extended input assignment lists.

Every extended input byte assigned in the control unit is identified by a "1" in the data word.

A "0" indicates that the extended input byte has not been assigned. 16 extended input bytes per data word are identified as assigned or not assigned.

Data word 65 for assignment list of the extended input byte 15 ... 0

DEFW K0000000000000000B

Example: K0000000000000001B = EI byte 0/EI0.0 to EI0.7 present

Data word 66 for assignment list of the extended input byte 31 ... 16

DEFW K0000000000000000B

Example: see data word 65

Data word 67 for assignment list of the extended input byte 47 ... 32

DEFW K0000000000000000B

Example: see data word 65

Data word 68 for assignment list of the extended input byte 63 ... 48

DEFW K0000000000000000B

Example: see data word 65

.
.
.

Data word 80 for assignment list of the extended input byte 255 ... 240

DEFW K0000000000000000B

Example: see data word 65

Data words 81 to 96

Data words 81 to 96 are intended for the definition of peripheral extended output assignment lists.

Every extended output byte assigned in the control unit is identified by a “1” in the data word.

A “0” indicates that the extended output byte has not been assigned. 16 extended output bytes per data word are identified as assigned or not assigned.

Data word 81 for assignment list of the extended output byte **15 ... 0**

DEFW K0000000000000000B

Example: K0000000000000001B = EO byte 0/EO0.0 to EO0.7 present

Data word 82 for assignment list of the extended output byte **31 ... 16**

DEFW K0000000000000000B

Example: see data word 81

Data word 83 for assignment list of the extended output byte **47 ... 32**

DEFW K0000000000000000B

Example: see data word 81

Data word 84 for assignment list of the extended output byte **63 ... 48**

DEFW K0000000000000000B

Example: see data word 81

.
.
.

Data word 96 for assignment list of the extended output byte **255 ... 240**

DEFW K0000000000000000B

Example: see data word 81

from data word 96

The contents of data words from 96 onward are NC-specific and are described in detail in the ICL project planning manual (1070 073 741).

2.6 Program execution

Within the available organisation modules, the user can determine the start-up performance and modify other functions.

Criteria such as error recognition, interrupt inputs and timing periods during program execution result in an automatic call of the corresponding organisation module.

Program execution may be:

- cyclic
- peripheral interrupt controlled
- time-controlled

The following overview shows the criteria which determine the organisation modules for the respective type of program execution.

No of OM's (29)		OM number	Explanation		
1	OM1	1	cyclic execution		
1	OM2	2	initialisation table		
2	OM5	5	cold start with partial / without retention of memory		
	OM7	7	restart with partial / without retention of memory		
1	OM9	9	error recognition e.g. – cycle time exceeded – module nesting depth > 64 – invalid address – call of non-existent module etc.		
			Peripheral interrupt controlled execution	interrupt inputs	Priority
4	OM10	10		EI1.0	highest
	OM11	11		EI1.1	
	OM12	12		EI1.2	
	OM13	13		EI1.3	lowest
			time-controlled execution	time value	Priority
4	OM18	18	4 values can be preset in OM2 and modified during program execution.	time value 1	highest
	OM19	19		time value 2	
	OM20	20		time value 3	
	OM21	21		time value 4	lowest
			library modules	Usable in conjunction with high-level language	
16	OM48	48			
	OM49	49			
	OM62	62			
	OM63	63			

Specifications in the organisation modules

Priorities of the interrupt groups and specified timers

The following order of priority applies to interrupt groups and fixed timers:

- peripheral interrupts
- time-controlled execution

No time-controlled execution is permitted during the execution of the peripheral interrupt OM.

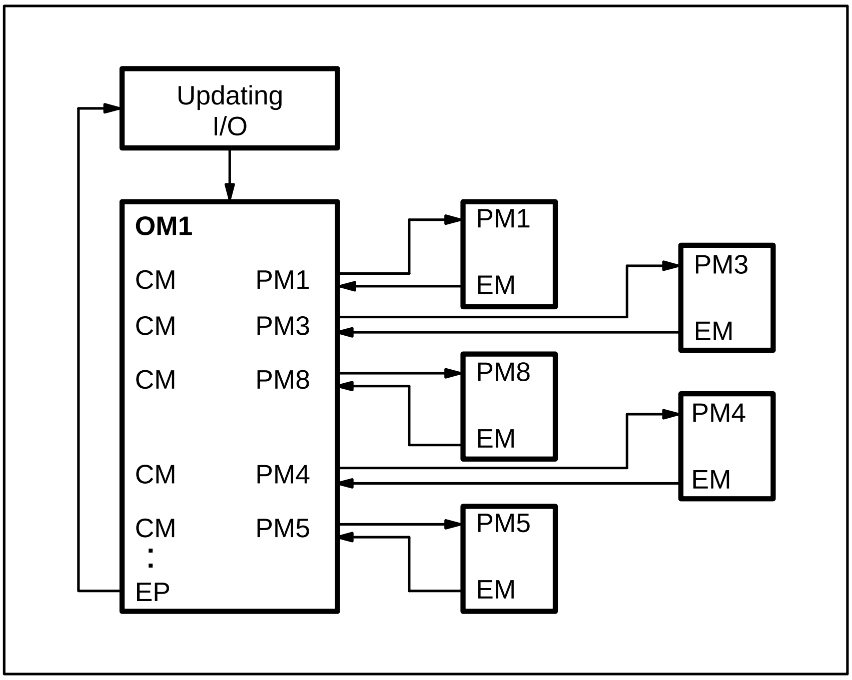
Peripheral interrupt controlled OMs can be executed while time-controlled OMs are being executed.

Priorities within an interrupt group

In the case of simultaneous interrupts within an interrupt group, the one with the lowest OM number has the highest priority.

2.6.1 Cyclic program execution

The modules of the user program are executed according to the sequence specified in the organisation module OM1. OM1 is automatically called after the start-up phase.



Example of cyclic program execution

2.6.2 Peripheral interrupt controlled program execution

General

In the event of an interrupt, cyclic execution is interrupted after execution of the command currently being processed and an organisation module assigned to the interrupt input is started. The user can program the desired reaction to the interrupt in this OM. Then cyclic program execution is restarted at the point of interruption.

Only the program status (flags) is recovered in the case of an interrupt. If register contents are changed in the interrupt OM, these must be recovered by the user himself.

All interrupts are stored in an interrupt register. There is a register for each interrupt group. One bit corresponds to one interrupt in these registers. When the OM is started the corresponding interrupt is automatically reset.

The commands **LAI** and **RAI** can be used to read and reset the stored interrupts.

Interrupts of a group can generally be enabled or disabled with the commands EAI (Enable Interrupts) and DAI (Disable Interrupt).

If an interrupt OM is to be executed only once, although there is continual change at the interrupt input, the user must disable the interrupt in the interrupt OM using the interrupt mask.

☞ **In the event of an interrupt, the register and scratch flag contents must be recovered by the user if required.**

Interrupt mask

There is one interrupt mask for each interrupt group.

Every bit in the mask is allocated to one specific interrupt. The individual interrupts within a group can be enabled or disabled by setting the corresponding bit in the relevant mask to **“1” or “0”.** The user can read or write this mask with the commands **“TIM” and “LIM”.**

If a bit is set in this mask, the corresponding interrupt is enabled.

The suppression of interrupts has no effect on their storage. When an interrupt arises, the OM will only be called if the interrupt has been enabled and has not been suppressed. If the OM has not been programmed, OM9 will be called for error treatment, and the module may be stopped.

All existing interrupts are deleted during start-up or the transition from **stop** to **run.** The interrupts are all disabled and suppressed and the user must enable them specifically.

All peripheral interrupts are deleted during a start with retention of memory. The mask and the enable/disable status return to the status before the interrupt.

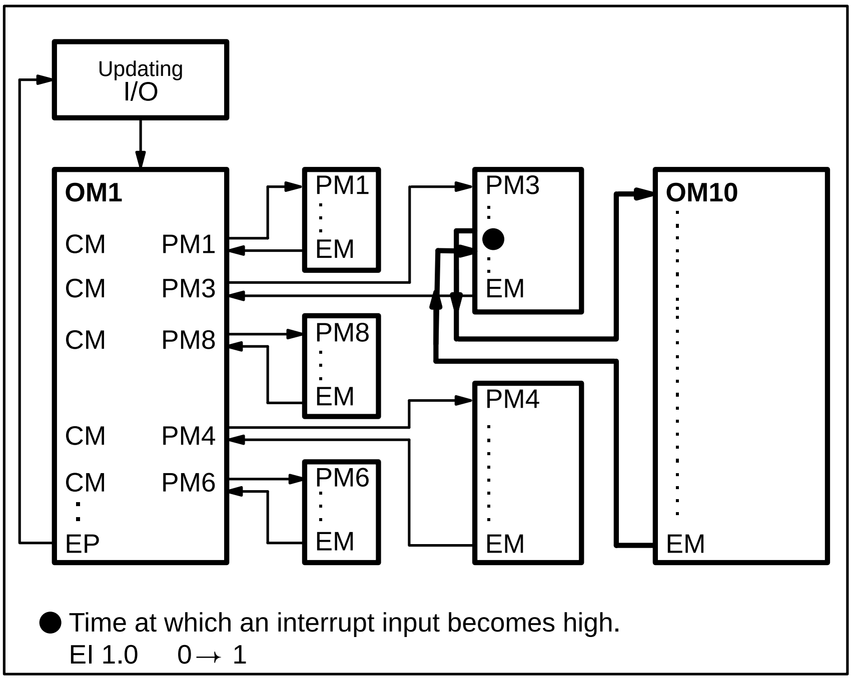
All interrupts are disabled during the execution of a start-up OM. Enabled interrupts are also recognised and processed in the I/O state.

Peripheral interrupts

A peripheral interrupt is triggered by a positive edge on one of the interrupt inputs EI1.0-EI1.3 of the interrupt module.

☞ **For as long as inputs or outputs (II, IO, EI, EO) are directly accessed, all peripheral interrupts must be disabled!**

Disabled interrupts have no temporal effect on program execution.



Example of peripheral interrupt controlled execution

☞ **Interrupts may not be nested, i.e. interrupt processing must not be interrupted by a further interrupt!**
Interrupt modules must be concluded with “EM”!

Example

```
EAI   PI      ; Enable peripheral interrupts
L     K1,A
TM    A,PI    ; Enable the peripheral interrupt with the highest value
              ; in the interrupt mask register
```

The above sequence of commands may be carried out only once, e.g. in start-up OM or in OM1.

A response to this interrupt of the input card (II 24) can be programmed in the interrupt OM10.

OM10:

```
L     M10,A
INC   A,1
T     A, M10
BE            ; The module must be terminated with EM.
```

2.6.3 Time-controlled program execution

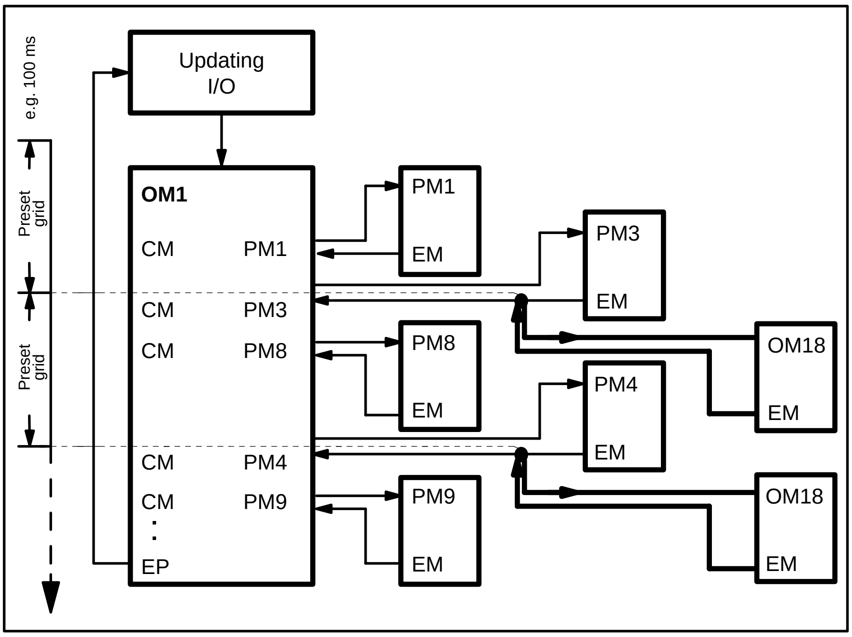
This function is initiated by internal timers, if these were defined through user commands in time values. Time values may be preset in OM2 or by means of the program by a description in system range **S**. Time values are activated after the next I/O cycle. To initiate time-controlled program execution, cyclic program execution is interrupted during operation of the corresponding timer and restarted at the point of interruption after execution (but only in the case of a module change).

Using PLC commands, time-controlled execution can be disabled, enabled, reset or suppressed for all timer modules or individually.

The timer module with the lowest number has priority over the timer module with the highest number.

Selectable timers: 10 ms x (1 - 65536)

☞ **The call of a timer OM does not disable other time-controlled processes.**
The time value must be preset with “0” for timer OMs which are not present.



Example of time-controlled program execution

Your notes:

3 Operation list

3.1 Structure of control statements

Control statements are executed in accordance with DIN 19239. They comprise an operation part and an operand part. However, the control statement may also consist of the operation part only, e.g. left bracket **“(”,** end of program **“PE”.**

Operation part

The operation part contains a maximum of 4 characters as a mnemonic short-hand command. It is divided into operator (OPR) and attribute. The attribute deSignates the data format.

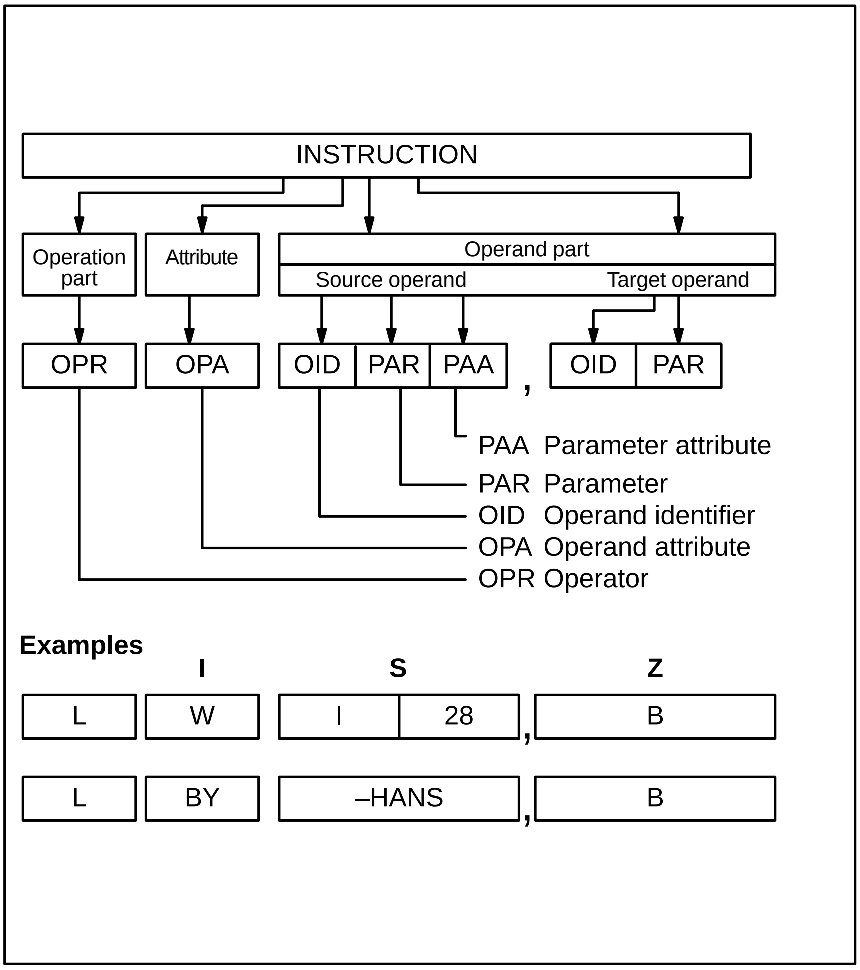
Operand part

The operand part contains the data necessary for the execution of an instruction. The format of operands can be symbolic or absolute.

In the case of absolute format, the operand part (depending on the Operation part) comprises one or two operands and one operand attribute (OPA). The operand attribute specifies the data format.

Each of these operands consists of an operand identifier (OID) and a parameter. The parameter (PAR) may be a bit, byte or word address.

In the case of symbolic format, the operand is marked by a preceding hyphen “–” and may comprise up to eight characters (letters and/or digits).



Structure of control statements

3.2 Operands

The following operands are available:

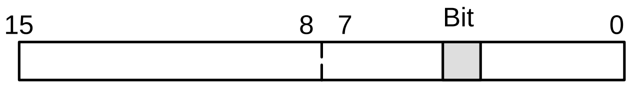
I – Input
O – Output
M – Marker
T – Timer
C – Counter
SM – Special marker
II – Interface input
EI – Extended input
IO – Interface output
EO – Extended output
DB – Data buffer
S – System range
P – Parameter
DF – Data field
K – Constant
D – Data word value
DX – Extended data word value
PM – Program module
DM – Data module
Pn – Number as parameter (n = 0 to 62)
TI – Time interrupt
PI – Peripheral interrupt
R – Register
[R] – Index register
I[R] – Input (indirect register)
O[R] – Output (indirect register)
M[R] – Marker (indirect register)
T[R] – Timer (indirect register)
C[R] – Counter (indirect register)
SM[R]– Special marker (indirect register)
II[R] – Interface input (indirect register)
EI[R] – Extended input (indirect register)
IO[R] – Interface output (indirect register)
EO[R]– Extended output (indirect register)
DB[R]– Data buffer (indirect register)
S[R] – System range (indirect register)
DF[R]– Data field (indirect register)
D[R] – Data module value (indirect register)
DX[R]– Extended data module value (indirect register)
PM[R]– Program module (indirect register)
DM[R]– Data module (indirect register)

In the above list, “R” is replaced with the register identifier “A”, “B”, “C”, “D”!

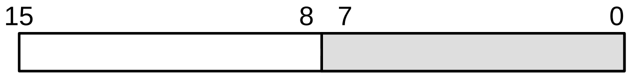
3.3 Data formats and register structure

Data formats

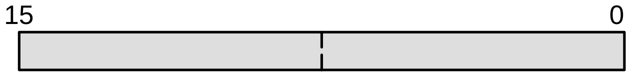
BIT = B



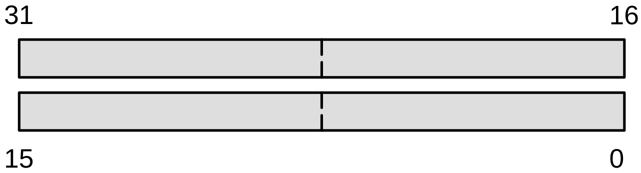
BYTE = BY



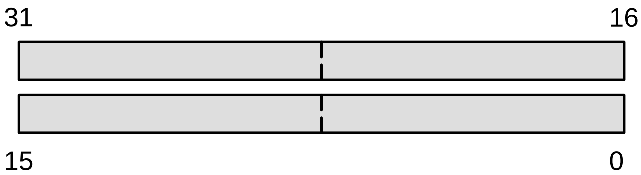
WORD = W



Double word = DW



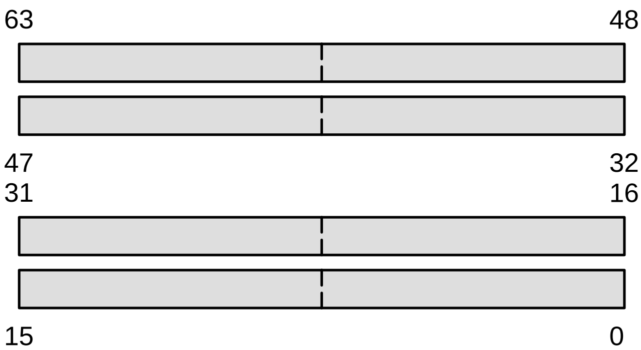
Floating point in double word format (in preparation)



The bits of the double word are divided into:

Mantissa = bit 0-22
Exponent = bit 23-30
Sign = bit 31

Floating point in four word format according to IEEE standard P 754 (in preparation)



The bits are divided into:

Mantissa = bit 0-51
Exponent = bit 52-62
Sign = bit 63

Data formats

Register structure

The control unit has four working registers which can be operated by bit, byte, word and double word; with bytes, the right byte is operated and with words, the right word is operated.

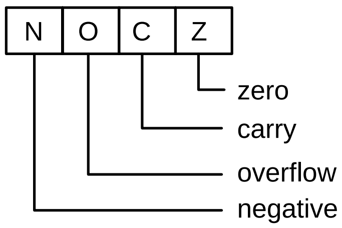
Working register A	Double word
Working register B	Double word
Working register C	Double word
Working register D	Double word

For operations which exceed the 32 bit format, fixed register pairs are formed from the individual registers.

Example

Register pair A + B	Double word + Double word
Register pair C + D	Double word + Double word

Status bit



Register structure

3.4 Types of addressing

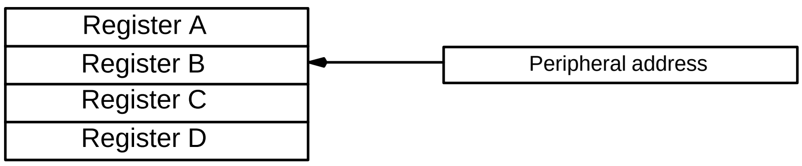
The following types of addressing are possible:

Direct addressing of all operands suitable for absolute addressing

t = timer and counter status
a = timer and counter actual value

	Operand group
Bit addressable	I, O, M, SM, T/C, D, DX, DB, DF, S, I[A], O[A], M[A] etc.
Byte/word/double word readable	I, O, M, SM, T/C, DB, DF, D, DX, S, II, EI, I[A], O[A] etc.
Byte/word/double word writeable	O, M, DB, DF, D, DX, S, IO, EO, T/C (with specific commands only)

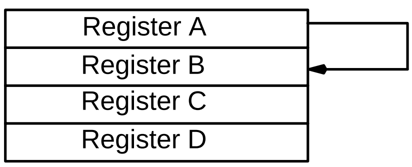
e.g. L BY I15,B



The contents of a peripheral address are loaded into a register, or the contents of a register are transferred to a peripheral address.

Register addressing

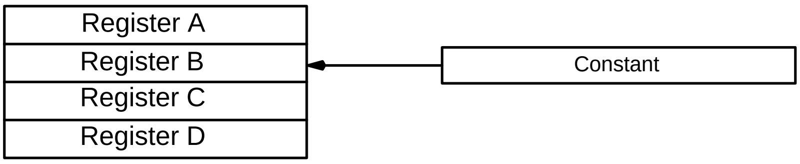
e.g. L W A,B



The contents of a register are loaded into another register.

Direct addressing

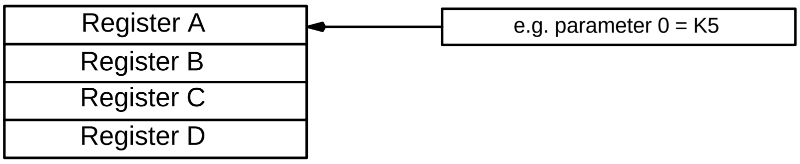
e.g. L W K1234H,B



A constant specified in the command is loaded into a register.

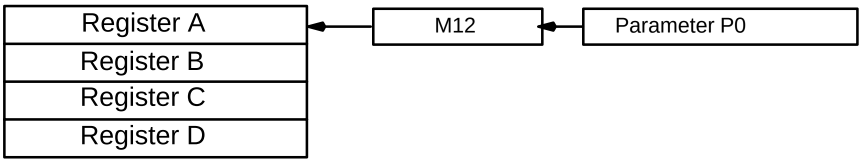
Parameter addressing – 2 cases

e.g. L W P0,A ;**case 1** (parameter 0 = constant K5)



Parameter 0 is loaded into register A.

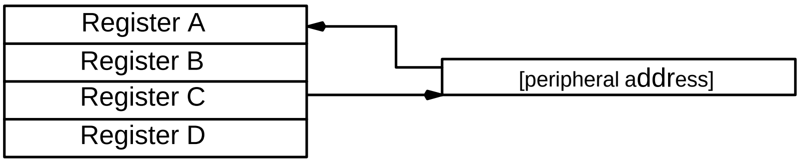
e.g. L W P0,A ;**case 2** (parameter 0 = address M12)



Register A is loaded with the marker contents whose address was allocated to P0.

Register-indirect addressing

e.g. L W M [C] ,A



Register A is loaded with the marker contents whose address is in register C.

Absolute address allocation with register-indirect addressing

Each of the four working registers can be used as an index register. Depending on the command, all operands which are permissible as a direct address can be operated. The number of the corresponding actual value must be entered in the index register for timer/counter start and timer/counter commands. In the case of illegal command and operand combinations, programs are aborted with an address error message.
With indirect addressing the operand range is identified by the operand prefix. The index register contains the operand number and address.

Module call command: structure of index register

Module number
31 … 0

Index register in hex:

from – to	0000H 01FFH	According to DM0 – DM512
from – to	0000H 03FFH	According to PM0 – PM1023

Bit commands:

structure of index register

Byte address	Bit addr.
31 … 3	2 1 0

Index register in hex

from – to	0000H 00FFH	Counter statuses C0 – C256
from – to	0000H 00FFH	Timer statuses T0 – T256
from – to	0000H 00FFH	Special markers SM0.0 –SM31.7/SM0D – SM255D
from – to	0000H 07FFH	Inputs I0.0 – I255.7 / I0D – I2047D
from – to	0000H 07FFH	Outputs O0.0 – O255.7 / O0D – O2047D
from – to	0000H BFFFH	Markers M0.0 – M6143.7 / M0D – M49151D
from – to	0000H 0FFFH	Data buffers DB0.0 – DB511.7 / DB0D – DB4095D
from – to	0000H 0FFFH	System range S0.0 – S511.7 / S0D – S4095D
from – to	0000H 0FFFH	Data word D0.0 – D511.7 / D0D – D4095D
from – to	0000H 0FFFH	extended data word DX0.0 – DX511.7 / DX0D – DX4095D
from – to	0000H 3FFFFH	Data field DF0.0 – DF32767.7 / DF0D – DF262143D

Byte, word and double word commands

Structure of index register

Byte addresss or timer/counter number	
31	0

Index register in hex

from – to	0000H 001FH	Special markers SM0.0–SM31.0
from – to	0000H 00FFH	Inputs I0.0–E255.7
from – to	0000H 00FFH	Outputs O0.0–A255.7
from – to	0000H 17FFH	Markers M0.0–M6143.7
from – to	0000H 01FFH	Data buffers DB0 – DB511
from – to	0000H 01FFH	System range S0–S511
from – to	0000H 00FFH	Counter actual value C0–C255
from – to	0000H 00FFH	Timer actual value T0–T255
from – to	0000H 01FFH	Data word D0–D511
from – to	0000H 01FFH	extended data word DX0 – DX511
from – to	0000H 00FFH	Interface inputs II0–II255
from – to	0000H 00FFH	Interface outputs IO0–IO255
from – to	0000H 00FFH	Extended inputs EI0–EI255
from – to	0000H 00FFH	Extended outputs EO0–EO255
from – to	0000H 7FFFH	Data field DF0–DF32767

Some examples of indirect addressing of bit addresses

Counter status “0” or “1”

```
L     DW    K0001H,A    ;Load index address A with address
                        ;of counter C1
A     B     Z[A]
=           M100.0
```

Timer status “0” or “1”

```
L     DW    K000AH,A    ;Load index address A with address
                        ;of timer T10
A     B     T[A]
=           M100.2
```

Query I0.0 to I3.7 for 1

```
L     DW    K0000H,A    ;Load index register A with address
                        ;I0.0
L     DW    K32D,B      ;Number of inputs 32

– start                 ;Loop start

A     B     I[A]        ;indirect binary query
JPCI        –false      ;abort at “0”
INC         A,1         ;Increment index register

DEC         B,1         ;Number 1
JPN         – start     ;32 queries made?

–false
.
.
.
```

Set indirect bit

```
A           SM31.1      ;logic 1
L     DW    K0040H,A    ;Load index register A with address
                        ;O8.0
S     B     O[A]
```

Some examples of indirect addressing of word addresses

Load counter actual values
Index addresses are counter numbers

```
L         K3,B

A         I0.0
SC        B,C8

A         I0.1
CD        C8

L    DW   K0000H,A
L    W    C[A],B     ;Load counter actual value C0

L    DW   K0001H,C
L    W    C[C],A     ;Load counter actual value C1
```

Query timer status “0” or “1”
Index registers are timer numbers

```
L         K100.3,D

A         I0.3
SPE       D,T0

L    DW   K0010,A    ;Timer status T10
A         T[A]

L    DW   K0011H,C   ;Timer status T11
A         T[C]
```

Counter actual values, contents in word format

```
L    DW   K2H,A      ;Load counter actual value C2
L         C[A],C     ;to C

L    DW   K8H,A      ;Counter actual value C8
L         C[A],B
```

Timer actual values, contents in word format

```
L    DW   K0000H,A
L         T[A],A     ;Timer actual value T0

L    DW   K000AH,A   ;Timer actual value T10
L         T[A],A
```

Indirect module call:
Program module:

```
L    DW   K0001H,D     ;Index address program module PM1
CM        PM[D]
```

1st data module:

```
L    DW   K0002H,C     ;Index address data module DM2
CM        DM[C]
L         D10,A
```

2nd data module:

```
L    DW   K0003H,C     ;Index address data module DM3
CX        DM[C]
L         DX10,A
```

Indirect addressing for timer/counter start or timer/counter commands

```
L    DW   K2H,A        ;Index address counter 2
L         K6,B         ;Counter value 6

A         I0.5
SC        B,C[A]

A         I0.6
CD        C[A]

L    DW   K3H,A        ;Index address timer circuit 3
L         K100.1,B     ;Time value 10 s

A         I0.5
SPE       B,T[A]

EP
```

3.5 Address ranges

Operand	Byte address	No.	Bit address
I/O	0– 255		0.0– 255.7
II/IO	0– 255		0.0– 255.7
EI/EO	0– 255		0.0– 255.7
M	0–6143		0.0–6143.7
D	0– 511		0.0–511.7
T		0–255D	
C		0–255D	
SM	0– 31		0.0– 31.7
DB	0– 511		0.0–511.7
DF	0–32767		0.0–32676.7
P		0– 62D	
S	0– 511		0.0–511.7

The numbers of timers, counters and parameters may be used for bit commands, word commands and timer/counter commands.

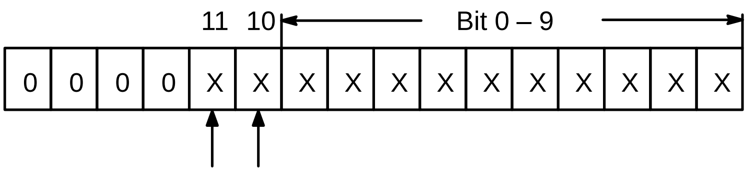
3.6 Representation of word constants

Basically, all representations apply also to data widths of 8 and 32 bits.

K0000000000000000B – K1111111111111111B	(Binary value)
K000000 O – K177777 O	(Octal value)
K00000D – K65535D	(Decimal value)
K0000H – KFFFFH	(Hexadecimal word)
K00000000H – KFFFFFFFFH	(Hexadecimal DW)
K0/0,A – K255/255,255/255,A	(4 bytes at a time)
K'ABCD'	(up to 4 ASCII characters owing to double word)
K0.R – K1023.R	(timer value)

☞ **Timer values are not entered directly, but with the multiplier "R" (grid); see below.**

3.7 Time format



0	0	=	10	ms	Grid 0
0	1	=	100	ms	Grid 1
1	0	=	1	s	Grid 2
1	1	=	10	s	Grid 3

Bits 0 to 9 = value entry
Bits 10 and 11 = grid

A constant entered in this format is loaded into the register.

3.8 Key to the special markers

SM14.0 – SM15.7 =	Extended error code (see SM29.6)
SM16.0 – SM17.7 =	reserved
SM18.0 – SM19.7 =	reserved
SM20.0 =	Initialising pulse
SM20.1 =	reserved
SM20.2 =	Flashing marker (2 Hz)
SM20.3 =	Lock output states
SM20.4 =	Fixing marker
SM20.5 =	reserved
SM20.6 =	reserved
SM20.7 =	Reload initialising pulse after power on or program reload
SM21.0 – SM21.7 =	reserved
SM22.0 – 23.7 =	Actual cycle time counter with factor 1 ms. The SM is refreshed cyclically every 40 ms (in preparation) or in the I/O, or the event of an error.
SM24.0 – 25.7 =	Max. cycle time with factor 1 ms. The cycle time is defined between OM1 start and the next OM1 start. Refresh before OM1 start. Reset with **stop** after **run change-over.**
SM26.0 – SM27.7 =	Min. cycle time with factor 1 ms. Refresh before OM1 start. Reset with **stop** after **run change-over.**
SM28.0 – SM29.7 =	Error word
SM28.0 =	reserved
SM28.1 =	Parameter error
SM28.2 =	Call of inexistent module
SM28.3 =	Module stack error
SM28.4 =	Application stack underflow
SM28.5 =	Application stack overflow
SM28.6 =	reserved
SM28.7 =	Direct access blocked
SM29.0 =	Write access prohibited
SM29.1 =	Opcode error
SM29.2 =	reserved
SM29.3 =	reserved
SM29.4 =	reserved
SM29.5 =	reserved
SM29.6 =	Identifier for error word 14
SM29.7 =	Cycle time error
SM30.0 – SM30.7 =	Internally used and not accessible to user

SM30.3 – fixed “0”

SM31.0 =	reserved
SM31.1 =	fixed ”1”
SM31.2 =	reserved
SM31.3 =	Carry *
SM31.4 =	reserved
SM31.5 =	Overflow *
SM31.6 =	Negative **
SM31.7 =	Zero **

* These special markers are generally not affected by carry and overflow flags.
** The corresponding flags are mapped to this special marker only with the command “CPL accumulator, accumulator”.

Error codes special marker SM14:

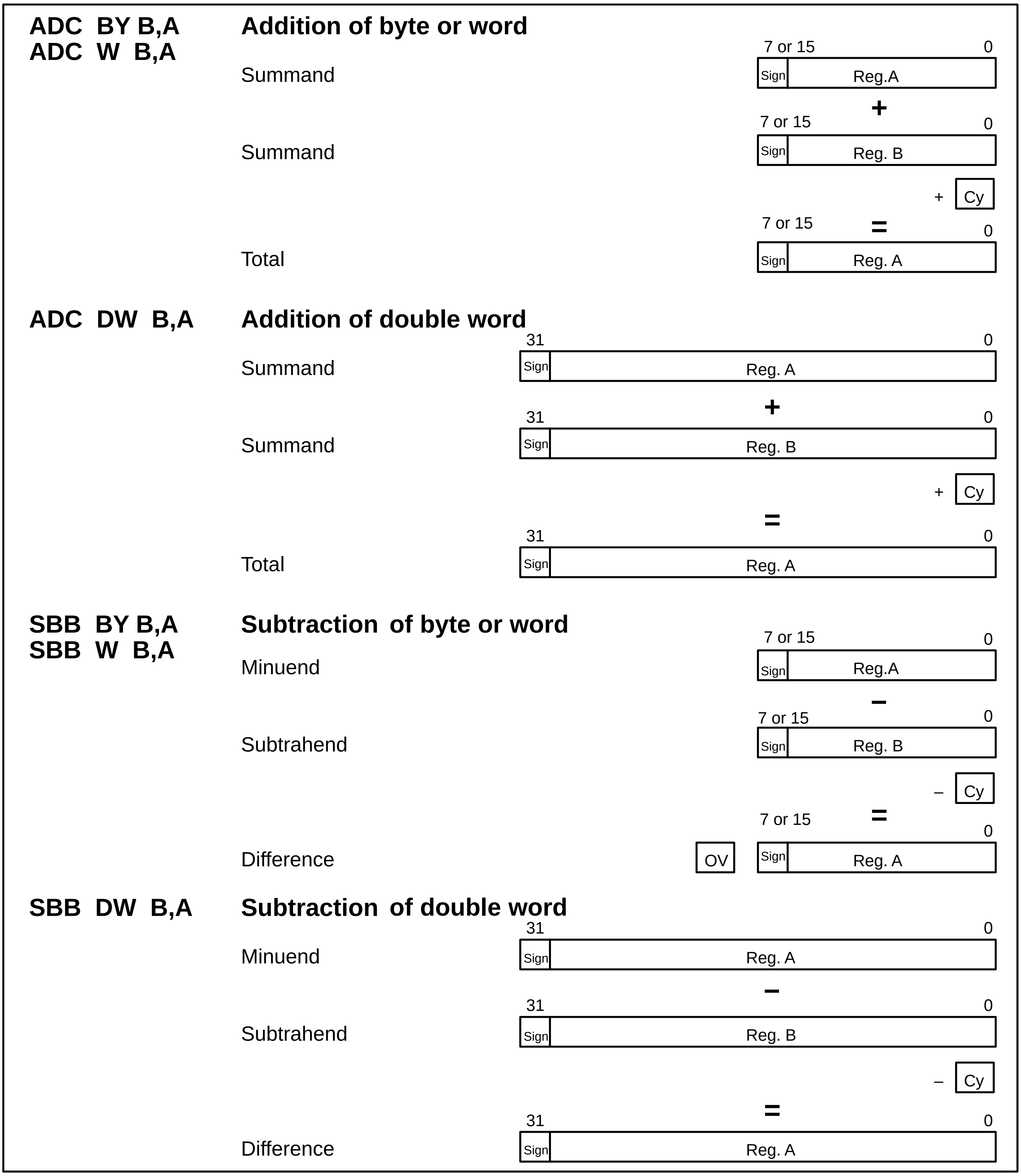
011H	Cycle time exceeded in I/O cycle
012H	Cycle time exceeded in program
016H	Block stack overflow: occurs if the module nesting depth is greater than 63
01aH	Opcode error
01bH	Parameter error: during access to a parameter to be read, either no parameter line was found or the parameter found could not be asSigned to the command (e.g. bit address in the case of a word command, image address in the case of a module call, etc.).
01cH	Address error: access to an invalid address. Possible causes: – transfer to a constant – parameter line read in as a command line – transfer to a timer/counter actual value
01dH	Area exceeded: access to a user memory which does not exist.
01eH	Module does not exist: during a module call command a module was called which is not in the reference list.
01fH	HOLD command
020H	Division by 0
021H	Control unit in STOP status
022H	Battery warning
023H	NC fault
024H	Internal system error
025H	Hardware error
026H	User error message from “C”
027H	User warning from “C”
028H	Reentrant module call
029H	Reference list error
02aH	CAN error
02bH	Internal communication error
02cH	APS error message
02dH	APS warning
020H	No PLC program

3.9 Key to the system range

The following initialisation values are strored in the system range

S1/S0	initialisation flags such as DW2 in OM2
S3/S2	reserved
S5/S4	error flags such as DW4 in OM2
S7/S6	maximum cycle time
S9/S8	number of the data module to be copied into the data buffer
S11/S10	number of the first retentive timer
S13/S12	number of the first retentive counter
S15/S14	number of the first retentive marker
S17/S16	first retentive address in data area
S19/S18	time value of timer module OM18
S21/S20	time value of timer module OM19
S23/S22	time value of timer module OM20
S25/S24	time value of timer module OM21
S27/S26	reserved
S29/S28	reserved
S31/S30	reserved
S33/S32	reserved
S121/S122	module number of the ICL 700
S123 to S511	reserved

3.10 Addition, subtraction, multiplication and division formats



Addition and subtraction formats

MUL BY B,A
MUL W B,A

Multiplication of byte or word

Multiplicand

7 or 15	Z	0
Sign	Reg.A	

Multiplier

7 or 15	Q	0
Sign	Reg. B	

Product

15 or 31	Z	0
Sign	Reg. A	

MUL DW B,A

Multiplication of double word

Multiplicand

31	Z	0
Sign	Reg. A	

Multiplier

31	Q	0
Sign	Reg. B	

31		Z + 1	0	31		Z	0
Sign	Reg. B	HIGH WORD		Reg. A		LOW WORD	

Product Double word

DIV BY B,A
DIV W B,A

Division of byte or word

Dividend

Sign 15 or 31	Z	0
Sign	Reg. A	

Divisor

Sign 7 or 15	Q	0
Sign	Reg. B	

Quotient, remainder

15 or 31		8 or 16	7 or 15	Z	0
Sign	remainder Reg. A		Sign	Quotient	

Note: **Pay attention to the Sign with division!**

DIV DW C,A

Division of double word

31		Z + 1	0	31		Z	0
Sign	HIGH WORD	Reg. B		LOW WORD		Reg. A	

Dividend Double word

Divisor

31	Q	0
Sign	Reg. C	

31		Z + 1	0	31		Z	0
Sign	Rest	Reg. B		Sign	Quotient	Reg. A	

Double word

Note: **Pay attention to the Sign with division!**

Multiplication and division formats

3.11 Instruction set

The following tables contain information on ICL700 commands.

The following abbreviations are used in the table in addition to the operands already known:

S	–	Source operand
Z	–	Destination operand / Zero
RES	–	Result
O	–	Overflow
C	–	Carry
N	–	Negative status of MSB
B	–	Bit
BY	–	Byte
W	–	Word
OPT	–	Operation
OPR	–	Operator
OPA	–	Operand attribute
R	–	Register (A, B, C or D)
[R]	–	Indirect register (A, B, C or D)
SY	–	symbolic *

* The symbols indicate places where symbols are permitted. The execution time corresponds to the operand to which the symbol was allocated.

Note on the flags:
N flag corresponds to the “MSB”! This results in differences to the previous control units with regard to flag behaviour:

			CL500:		ICL700:	
L	BY	K77H,A		A=127		A=127
INC	BY	A,1	O	A=128	O,N	A=128
SPP		–LABEL1		fulfilled		fulfilled
.						
.						
–LABEL1						
INC	BY	A,1	N	A=129	N	A=129
SPP		–LABEL2	N	not fulfilled		

OPT OPR	1.OPD. 2.OPD I S Z OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE (TARGET WITH S, R,=)	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Bit commands																					
A	B	I0.0	●			●		I, O, M, T, C, SM, SY D, DX, DB, DF, S		●					●						AND operation, query for signal status 1
AN	B	I0.0	●			●		I, O, M, T, C, SM, SY D, DX, DB, DF, S		●					●						AND operation, query for signal status 0
O	B	I0.0				●		I, O, M, T, C, SM, SY D, DX, DB, DF, S		●					●						OR operation, query for signal status 1
ON	B	I0.0				●		I, O, M, T, C, SM, SY D, DX, DB, DF, S		●					●						OR operation, query for signal status 0
S	B	O0.0		●	1		●	O, M, SY, SM D, DX, DB, DF, S		●											RES = 0 Set bit conditionally RES = 1
R	B	O0.0		●	1		●	O, M, SY, SM D, DX, DB, DF, S		●											RES = 0 Reset bit conditionally RES = 1
=	B	O0.0		●	1		●	O, M, SY, SM D, DX, DB, DF, S		●											Status unchanged Result allocation Status changes
A A A	B B B	P0 I.0 I[A]	● ● ●			● ● ●		Parameter Register bit Prefix as direct operand			● ● ●		● ● ●	● ● ●	● ● ●						AND operation, query for signal status 1. Register: register bit from 0–31
AN	B	P0	● ● ●			● ● ●		Parameter Register bit Prefix as direct operand			● ● ●		● ● ●	● ● ●	● ● ●						AND operation, query for signal status 0. Register: register bit from 0–31
O	B	P0	● ● ●			● ● ●		Parameter Register bit Prefix as direct operand			● ● ●		● ● ●	● ● ●	● ● ●						OR operation, query for signal status 1. Register: register bit from 0–31
ON	B	P0	● ● ●			● ● ●		Parameter Register bit Prefix as direct operand			● ● ●		● ● ●	● ● ●	● ● ●						OR operation, query for signal status 0. Register: register bit from 0–31

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE (TARGET WITH S, R,=)	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Bit commands																					
S S	B B	P0 I.0		•	1		•	Parameter Register bit Prefix as direct operand			•		•	•							RES = 0 Set bit conditionally RES = 1
R	B	P0		•	1		•	Parameter Register bit Prefix as direct operand			•		•	•							RES = 0 Reset bit conditionally REs = 1
=	B	P0		•	1		•	Parameter Register bit Prefix as direct operand			•		•	•							Result allocation
S	B	M [A]		•	1		•	[R]			•		•	•							RES = 0 Set bit conditionally RES = 1
R	B	M [A]		•	1		•	[R]			•		•	•							RES = 0 Reset bit conditionally RES = 1
=	B	M [A]		•	1		•	[R]			•		•	•							Result allocation
P	B	A.n						R			•						•				Check the register, bit n (n = 0–31) for status 1 (fulfilled: C = 1)
TSTZ	B	A.n						R			•						•				Check the register, bit n (n = 0–31) for status 0 (fulfilled: C = 1)

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
A	DW W BY	K0,A						K	R			●				0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	P0,A						P	R				●			0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	D0,A						D, DX	R	●						0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	II0,A						II, EI	R	●						0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	B,A						R	R		●					0	0	●	●		AND operation between source and target. Result in target.
A	DW W BY	M [B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		AND operation between source and target. Result in target.

BOSCH

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
AN	DW W BY	K0,A						K	R			●				0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	P0,A						P	R				●			0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	D0,A						D, DX	R	●						0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	II0,A						II, EI	R	●						0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	B,A						R	R		●					0	0	●	●		AND NOT operation between source and target. Result in target.
AN	DW W BY	M [B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		AND NOT operation between source and target. Result in target.

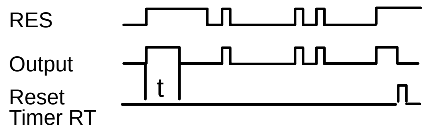
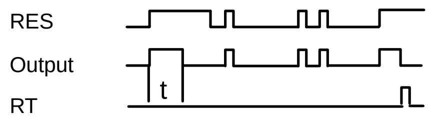
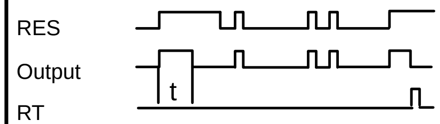
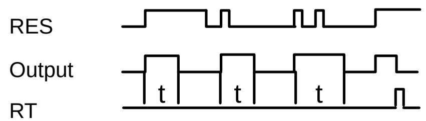
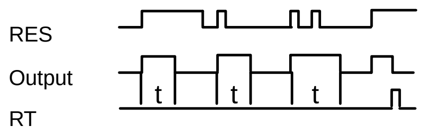
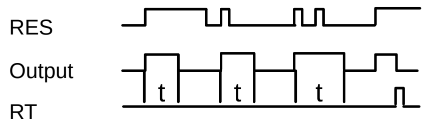
1070 073 737-103 (98.12) GB

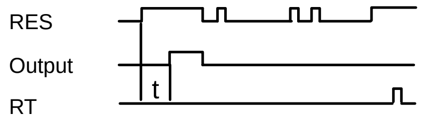
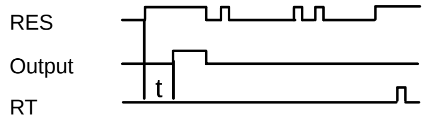
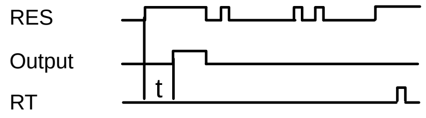
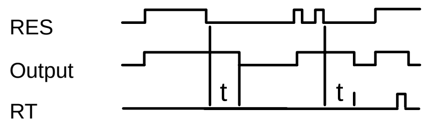
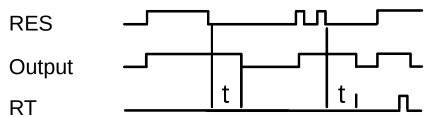
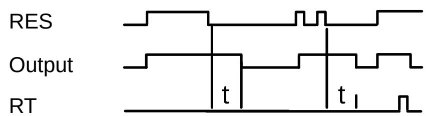
OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instructio	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
O	DW W BY	K0,A						K	R			●				0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	P0,A						P	R				●			0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	D0,A						D, DX	R	●						0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	II0,A						II, EI	R	●						0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	B,A						R	R		●					0	0	●	●		OR operation between source and target. Result in target.
O	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		OR operation between source and target. Result in target.

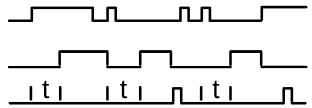
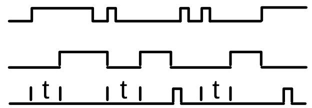
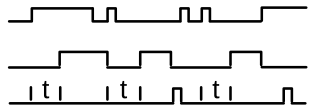
OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
ON	DW W BY	K0,A						K	R			●				0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	P0,A						P	R				●			0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	D0,A						D, DX	R	●						0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	II0,A						II, EI	R	●						0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	B,A						R	R		●					0	0	●	●		OR NOT operation between source and target. Result in target.
ON	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		OR NOT operation between source and target. Result in target.

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Qeury	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
XO	DW W BY	K0,A						K	R			●				0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	P0,A						P	R				●			0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	D0,A						D, DX	R	●						0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	II0,A						II, EI	R	●						0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	B,A						R	R		●					0	0	●	●		EXCL OR operation between source and target. Result in target.
XO	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		EXCL OR operation between source and target. Result in target.

OPT OPR	1.OPD. 2.OPD I OPA	S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Byte and word commands																					
XON	DW W BY	K0,A						K	R			●				0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	P0,A						P	R				●			0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	D0,A						D, DX	R	●						0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	II0,A						II, EI	R	●						0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	B,A						R	R		●					0	0	●	●		EXCL OR NOT operation between source and target. Result in target.
XON	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		0	0	●	●		EXCL OR NOT operation between source and target. Result in target.

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.		Comment
Timer commands																	
SP		A,T0			●	1 ↑ 0			R	T SY		●					Start timer as impulse 
SP		A,P0			●	1 ↑ 0			R	P		●					Start timer as impulse 
SP		A,T[B]			●	1 ↑ 0			R	[R]		●					Start timer as impulse 
SPE		A,T0			●	1 ↑ 0			R	T SY		●					Start timer as extended impulse 
SPE		A,P0			●	1 ↑ 0			R	P		●					Start timer as extended impulse 
SPE		A,T[B]			●	1 ↑ 0			R	[R]		●					Start timer as extended impulse 

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.		Comment
Timer commands																	
SR		A,T0			●	1 ↑ 0			R	T SY		●					Start timer as power-up delay 
SR		A,P0			●	1 ↑ 0			R	P		●					Start timer as power-up delay 
SR		A,T[B]			●	1 ↑ 0			R	[R]		●					Start timer as power-up delay 
SF		A,T0			●	1 ↑ 0			R	T SY		●					Start timer as power-off delay 
SF		A,P0			●	1 ↑ 0			R	P		●					Start timer as power-off delay 
SF		A,T[B]			●	1 ↑ 0			R	[R]		●					Start timer as power-off delay 

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.		Comment
Timer commands																	
SRE		A,T0			●	1 ↑ 0			R	T SY		●					Start timer as storable power-up delay RES Output RT 
SRE		A,P0			●	1 ↑ 0			R	P		●					Start timer as storable power-up delay RES Output RT 
SRE		A,T[B]			●	1 ↑ 0			R	[R]		●					Start timer as storable power-up delay RES Output RT 

Example:

L	K100.0,A	;	Load register A with (100 x 10 ms) 1000 ms
A	I0.0	;	Rising edge for timer start
SPE	A,T0	;	Start timer circuit T0 with 1000 ms

1070 073 737-103 (98.12) GB

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Timer commands																					
RT		T0		●	1			T, SY		●											with RES = 0 not executed Reset timer with RES =1 executed
TH		T0		●	1			T, SY		●											with RES = 0 not executed Stop timer with RES =1 executed
RT		P0		●	1			P					●								with RES = 0 not executed Reset timer with RES =1 executed
TH		P0		●	1			P					●								with RES = 0 not executed Stop timer with RES =1 executed
RT		T [A]		●	1			[R]						●							with RES = 0 not executed Reset timer with RES =1 executed
TH		T [A]		●	1			[R]						●							with RES = 0 not executed Stop timer with RES =1 executed

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.		Comment
Counter commands																
SC		A,C0		●	1 ↑ 0		●	R	Z SY		●					Load the counter with the value in the register.
SC		A,P0		●	1 ↑ 0		●	R	P		●					Load the counter addressed by the parameter with the value in the register.
SC		A. C[B]		●	1 ↑ 0		●	R	[R]		●					Load the indirectly addressed counter with the value in the register.
CV		C0		●	1 ↑ 0		●	C, SY		●						Count up
CV		P0		●	1 ↑ 0		●	P					●			Count up
CV		C[A]		●	1 ↑ 0		●	[R]						●		Count up

OPT OPR	1.OPD. I OPA	S OID PAR	2.OPD Z	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.		Comment
Counter commands																	
CD		C0			●	↑ (0→1)		●	C, SY		●						Count down
CD		P0			●	↑ (0→1)		●	P					●			Count down
CD		C[A]			●	↑ (0→1)		●	[R]						●		Count down
RC		C0			●	1		●	C, SY		●						Reset counter
RC		P0			●	1		●	P					●			Reset counter
RC		C[A]			●	1		●	[R]						●		Reset counter

OPT	1.OPD. 2.OPD									ADDR TYPE					STATUS						
	I	S Z																			
OPR	OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	O	C	N	Z		Comment
Compare commands																					
CPL	DW W BY	A, A						R	R		●							●	●		Compare the contents of A and B The contents are dealt with as an integer and not as a two's complement.

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Compare commends																					
CPLA	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	•						•	•	•	•		General comparison of the contents of A and I0
CPLA	DW W BY	D0,A						D, DX	R	•						•	•	•	•		General comparison of the contents of A and D0
CPLA	DW W BY	II0,A						II, EI	R	•						•	•	•	•		General comparison of the contents of A and II0
CPLA	DW W BY	B,A						R	R		•					•	•	•	•		General comparison of the contents of A and B
CPLA	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					•		•	•	•	•		General comparison of the contents of A and [B]
CPLA	DW W BY	K0,A						K,	R			•				•	•	•	•		General comparison of the contents of A and K0
CPLA	DW W BY	P0,A						P	R				•			•	•	•	•		General comparison of the contents of A and P0

Note:

The comparison can be evaluated arithmetically and logically.

Arithmetic: The value is considered as a positive or negative two's complement number.

Logic: The value is considered as a positive integer.

Further information on comparison commands CPL and CPLA

General note on numerical representation

Positive number:	0H	to	7FH	(byte)
	0D	to	127D	(byte)
	0H	to	7FFFH	(word)
	0D	to	32767D	(word)
Negative number:	80H	to	FFH	(byte)
	–128D	to	–1D	(byte)
	8000H	to	FFFFH	(word)
	–32768D	to	–1D	(word)

Arithmetic: positive and negative numerical value representation

Numerical values from	0D	to	127D	(byte)
and from	–128D	to	–1D	(byte)
or				
from	0D	to	32767D	(word)
and from	–32768D	to	–1D	(word)

Logic: positive numerical value representation only

Numerical values from	0D	to	255D	(byte)
or				
from	0D	to	65535D	(word)

Formation of C, N, O, Z, AGR (arithmetically greater) and LGR (logically greater)

ADD B, A

A +	B =	Ext.	C	O	N	Z	AGR	LGR
+	+	– (+)	0	1	0	0	1	1
+	+	+	0	0	0	0	1	1
–	+	+	1	0	0	0	1	0
–	+	0	1	0	0	1	0	0
–	+	–	0	0	1	0	0	1
+	–	+	1	0	0	0	1	0
+	–	0	1	0	0	1	0	0
+	–	–	0	0	1	0	0	1
–	–	+ (–)	1	1	1	0	0	0
–	–	0 (–)	1	1	1	1	0	0
–	–	–	1	0	1	0	0	0

The Sign in brackets indicates the actual result.

SUB B, A; CPLA B, A

A	–	B	=	Ext.	Log. CPL A B	Arith. CPL A B	C	O	N	Z	AGR	LGR
+		+		–	<	<	1	0	1	0	0	0
+		+		0	=	=	0	0	0	1	0	0
+		+		+	>	>	0	0	0	0	1	1
–		+		+ (–)	>	<	0	1	1	0	0	1
–		+		–	>	<	0	0	1	0	0	1
+		–		+	<	>	1	0	0	0	1	0
+		–		– (+)	<	>	1	1	0	0	1	0
–		–		+	>	>	0	0	0	0	1	1
–		–		0	=	=	0	0	0	1	0	0
–		–		–	<	<	1	0	1	0	0	0

The Sign in brackets indicates the actual result.

Logic comparison evaluation

The flags C, Z, LGR are available for the logic comparison evaluation.

They are evaluated according to the following table:

A B:	C	Z	LGR
<	+1	0	0
<,=	*	*	+0
>	0	0	+1
>,=	+0	*	*
=	0	+1	0
#	*	+0	*

+: This flag gives clear information on the comparison.
* : This flag can have either value (0, 1).

Arithmetic comparison evaluation

The flags N, Z, AGR are available for the arithmetic comparison evaluation.

They are evaluated according to the following table:

A B:	N	Z	AGR
<	+1	0	0
<,=	*	*	+0
>	0	0	+1
>,=	+0	*	*
=	0	+1	0
#	*	+0	*

+: This flag gives clear information on the comparison.
* : This flag can have either value (0, 1).

Formation of C, N, O, Z, AGR (arithmetical greater) and LGR (logically greater)

CPL B, A:

A	– B =	Ext.	Log. CPL A B	Arith. CPL A B	C	O	N	Z	AGR	LGR
+	+	–	<	<	*	*	1	0	0	*
+	+	0	=	=	*	*	0	1	0	0
+	+	+	>	>	*	*	0	0	1	*
–	+	+	>	<	*	*	0	0	1	*
–	+	–	>	<	*	*	0	0	1	*
+	–	+	<	>	*	*	1	0	0	*
+	–	–	<	>	*	*	1	0	0	*
–	–	+	>	>	*	*	0	0	1	*
–	–	0	=	=	*	*	0	1	0	0
–	–	–	<	<	*	*	1	0	0	*

The Sign in brackets indicates the actual result.

* : The old value is retained

AGR and LGR with the CPL command

AGR and LGR cannot be evaluated with the CPL command. Only a logic evaluation as follows via N and Z flags is possible with this comparison.

A B:	N	Z
>	0	0
=	0	+1
<	+1	0
>,=	+0	*
#	*	+0

+: This flag gives clear information on the comparison.
* : This flag can have either value (0, 1).

Programming the individual comparison results with the special marker when using the CPL commands:

```
CPL     W     B.A
```

Less than

```
A             SM31.6      ; A < B
```

Less than-equal to

```
A             SM31.6      ; A < = B
O             SM31.7
```

Equal

```
A             SM31.7      ;  A = B
```

Not equal

```
AN            SM31.7      ;  A # B
```

Greater than-equal to

```
AN            SM31.6      ;  A > = B
```

Greater than

```
AN            SM31.6      ;  A > B
AN            SM31.7
```

OPT OPR	1.OPD. 2.OPD I S Z OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Code changes																					
BID	DW W BY	A						R			●					●	0	0	●		Change binary to decimal (BCD) With overflow, the overflow bit is set.
DEB	DW W BY	A						R			●					●	0	0	●		Change decimal to binary Wrong BCD coding sets the overflow bit.
Replace commands																					
SWAP	W DW	A A						R			●										Interchange the high byte and the low byte in the register. Also high word and low word.

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Stack commands																					
PUSH	DW	A						R			●										Write register to user stack and decrement the user stack address.
POP	DW	A						R			●										Increment the user stack aADDRess and read from the user stack.

Note: The stack area comprises 256 words

With underflow of the user stack, special marker SM28.4 is set.

With overflow of the user stack, special marker SM28.5 is set.

The user stack is deleted in the I/O state.

OPT OPR	1.OPD. 2.OPD I S Z OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	STATUS O	C	N	Z		Comment
Arithmetic commands																					
ADD	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	•						•	•	•	•		Fixed-point addition of integers with sign A + P → Result is in A
ADD	DW W BY	D0,A						D, DX	R	•						•	•	•	•		Fixed-point addition of integers with sign A + D → Result is in A
ADD	DW W BY	II0,A						II, EI	R	•						•	•	•	•		Fixed-point addition of integers with sign A + II → Result is in A
ADD	DW W BY	B,A						R	R		•					•	•	•	•		Fixed-point addition of integers with sign A + B → Result is in A
ADD	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					•		•	•	•	•		Fixed-point addition of integers with sign A + [B] → Result is in A
ADD	DW W BY	K0,A						K	R			•				•	•	•	•		Fixed-point addition of integers with sign A + K → Result is in A
ADD	DW W BY	P0,A						P	R				•			•	•	•	•		Fixed-point addition of integers with sign A + P → Result is in A

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Arithmetic commands																					
SUB	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	•						•	•	•	•		Fixed-point subtraction of integers with sign A – E → Result is in A
SUB	DW W BY	D0,A						D, DX	R	•						•	•	•	•		Fixed-point subtraction of integers with sign A – D → Result is in A
SUB	DW W BY	II0,A						II, EI	R	•						•	•	•	•		Fixed-point subtraction of integers with sign A – II → Result is in A
SUB	DW W BY	B,A						R	R		•					•	•	•	•		Fixed-point subtraction of integers with sign A – B → Result is in A
SUB	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					•		•	•	•	•		Fixed-point subtraction of integers with sign A – [B] → Result is in A
SUB	DW W BY	K0,A						K	R			•				•	•	•	•		Fixed-point subtraction of integers with sign A – K → Result is in A
SUB	DW W BY	P0,A						P	R				•			•	•	•	•		Fixed-point subtraction of integers with sign A – P → Result is in A

OPT	1.OPD. 2.OPD I	S Z								ADDR TYPE					STATUS						
OPR	OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	O	C	N	Z		Comment
Arithmetic commands																					
ADC	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						●	●	●	●		
ADC	DW W BY	D0,A						D, DX	R	●						●	●	●	●		
ADC	DW W BY	II0,A						II, EI	R	●						●	●	●	●		
ADC	DW W BY	B,A						R	R		●					●	●	●	●		Fixed-point addition allowing for the carry, e.g. multiword addition allowing for the carry from the low word addition.
ADC	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		●	●	●	●		
ADC	DW W BY	K0,A						K	R			●				●	●	●	●		
ADC	DW W BY	P0,A						P	R				●			●	●	●	●		

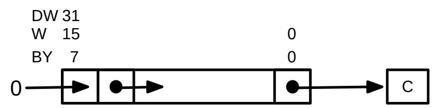
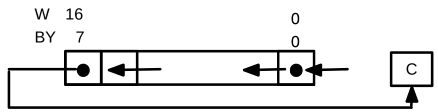
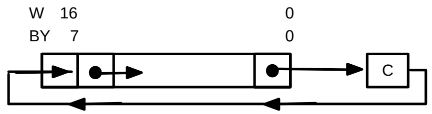
OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Arithmetic commands																					
SBB	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●						●	●	●	●		
SBB	DW W BY	D0,A						D, DX	R	●						●	●	●	●		
SBB	DW W BY	II0,A						II, EI	R	●						●	●	●	●		
SBB	DW W BY	B,A						R	R		●					●	●	●	●		Fixed-point subtraction allowing for the negative carry, e.g. multiword subtraction allowing for the loan bit from the low word addition.
SBB	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●		●	●	●	●		
SBB	DW W BY	K0,A						K	R			●				●	●	●	●		
SBB	DW W BY	P0,A						P	R				●			●	●	●	●		

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Arithmetic commands																					
MUL	DW W BY	A,B						R	R		•					0	0	•	•		Fixed-point multiplication of integers with sign B * A → Result is in A_B for DW
MUL	DW W BY	K10,A						K	R			•				0	0	•	•		Fixed-point multiplication of integers with sign B * A → Result is in A_B for DW
DIV	DW W BY	A,B						R	R		•					•	0	•	•		Fixed-point division of integers with sign C_B : A → Result is in C_B (R/Q) for DW B : A → Result is in B **Note:** 0 = 1 with – Division by 0 – Division overflow
DIV	DW W BY	K10,A						K	R			•				•	0	•	•		Fixed-point division of integers with sign B_A : K → Result is in B_A (R/Q) for DW A : K → Result is in A **Note:** 0 = 1 with: – Division by 0 – Division overflow

OPT OPR	1.OPD. 2.OPD I S Z OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Arithmetic commands																					
INC	DW W BY	A,n						R	R		●					●	●	●	●		Increase the contents of the source operand by “n” n = 0 – 127
INC	DW W BY	A,[C]						R	R		●					●	●	●	●		Increase the contents of the source operand by “n” n= 0 – 127 With n= 0 – repeat factor in register C
DEC	DW W BY	A,n						R	R		●					●	●	●	●		Decrease the contents of the source operand by “n”. n = 0 – 127
DEC	DW W BY	A,[C]						R	R		●					●	●	●	●		Decrease the contents of the source operand by “n” n= 0 – 127 With n= 0 – repeat factor in register C
TC	DW W BY	A						R			●					●	●	●	●		Complement the contents of the source operand (two’s complement)
N	DW W BY	A						R			●					0	0	●	●		Negate the contents of the source operand (one’s complement)
																					Note: The time is independent of “n”

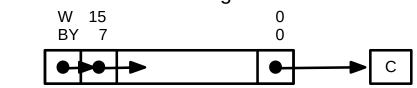
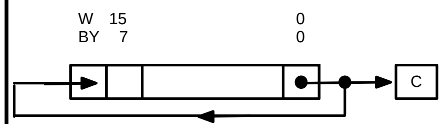
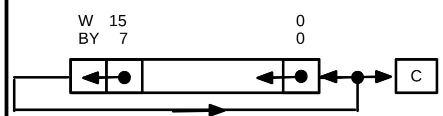
OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGENT	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Load commands																					
L	DW W BY	I0,A						I, O, M, T, C, SM DB, S (SY), DF	R	●											Load the source operand into the target register.
L	DW W BY	D0,A						D, DX	R	●											Load the source operand into the target register.
L	DW W BY	II0,A						II, EI	R	●											Load the source operand into the target register.
L	DW W BY	B,A						R	R		●										Load the source operand into the target register.
L	DW W BY	[B],A						[R] only with prefix Prefix as direct operand	R					●							Load the indirectly addressed source operand into the target register.
L	DW W BY	K0,A						K	R			●									Load the source operand into the target register.
L	DW W BY	P0,A						P	R				●								Load the source operand which is transfered with the parameter into the target register.

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	TARGET	SOURCE	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Transfer commands																					
T	DW W BY	A,O0						O, M, DB, S (SY), DF	R		●										Transfer from source register into the target operand
T	DW W BY	A,D2						D, DX	R		●										Transfer from source register into the target operand
T	DW W BY	A,IO						IO, EO	R		●										Transfer from source register into the target operand
T	DW W BY	B,A						R	R		●										Transfer from source register into the target operand
T	DW W BY	A,[B]						[R] only with prefix Prefix as direct operand	R		●										Transfer from source register into the indirectly addressed target operand
T	DW W BY	A,P0						P	R		●										Transfer from source register into the target operand

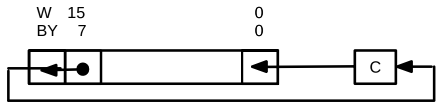
OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Shift commands															(all flags are influenced)						
SLR	DW W BY	A,n						R	R		●					●	●	●	●		Logic SHIFT to right 
SLR	DW W BY	A,[C]*						R	R		●					●	●	●	●		Logic SHIFT to right **As above:** The number of shift positions is in register C
SLL	DW W BY	A,n						R	R		●					●	●	●	●		Logic SHIFT to left 
SLL	DW W BY	A,[C]*						R	R		●					●	●	●	●		Logic SHIFT to left **As above:** The number of shift positions is in register C
RCR	DW W BY	A,n						R	R		●					●	●	●	●		Rotate right through CARRY 
RCR	DW W BY	A,[C]*						R	R		●					●	●	●	●		Rotate right through CARRY **As above:** The number of shift positions is in register C

Note: The time is independent of “n”. n = 0 – 131 With n = 0 the number of shift positions is in register C

* Only C register possible.

OPT OPR	1.OPD. 2.OPD I S Z OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Shift commands															(all flags are influenced)						
SAR	DW W BY	A,n						R	R		●					●	●	●	●		Arithmetic SHIFT to right  All liberated bits are filled with the sign
SAR	DW W BY	A,[C]						R	R		●					●	●	●	●		Arithmetic SHIFT to right **As above:** The number of shift positions is in the address pointer of [C]
ROR	DW W BY	A,n						R	R		●					●	●	●	●		Rotate to right 
ROR	DW W BY	A,[C]						R	R		●					●	●	●	●		Rotate to right **As above:** The number of shift positions is in the address pointer of [C]
ROL	DW W BY	A,n						R	R		●					●	●	●	●		Rotate to left 
ROL	DW W BY	A,[C]						R	R		●					●	●	●	●		Rotate to left **As above:** The number of shift positions is in the address pointer of [C]

Shift commands (all flags are influenced)

OPR	OPA	OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	O	C	N	Z		Comment
RCL	DW W BY	A,n						R	R		●					●	●	●	●		Rotate to right through CARRY 
RCL	DW W BY	A,[C]						R	R		●					●	●	●	●		Rotate to right through CARRY **As above:** The number of shift positions is in the address pointer of [C]

BOSCH

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Interrupt commands																					
LIM		TI,C						TI, PI	R	●											Load interrupt mask (TI, PI)
LAI		TI,C						TI, PI	R	●											Load the interrupt register of the interrupt group (TI, PI)

Note: LAI: – Incoming interrupts remain in interrupt register until execution; for example, when an interrupt is disabled it is possible to detect arriving interrupts. If interrupts are not reset, the stored interrupts will come after the “EAI” command.

TI: – Time-controlled processing

PI: – Peripheral interrupt-controlled processing

OPT OPR	1.OPD. 2.OPD I OPA	S Z OID PAR	First instruction	RES dependent	RES status	Query	Conclusion	SOURCE	TARGET	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z		Comment
Interrupt commands																					
TIM		A,TI						R	TI PI	●											Transfer the contents of the register into the corresponding interrupt mask (TI, PI) to disable or enable the interrupts.
RAI		A,TI						R	TI PI SI	●											Reset the interrupts in the interrupt register of the selected interrupt group (TI, PI)
EAI		PI						TI, PI		●											Enable interrupt group (TI, PI,)
DAI		PI						TI, PI		●											Disable interrupt group (TI, PI)
Parameter commands and other commands																					
Pi	DW W BY B	I0.0						I, O, M, T, C, SM, II EI, IO, EO, DB, S, SY, DF, K, D, DX, PM, DM		●											Parameter determined at param. module call i = 0 – 62
*		n																			Set help marker to examine program execution n = 0 – 63*
DEFW	W	K0						K													Instruction for system variables only in OM2

Note: n = 0 – 63* – Multiple setting of the same marker is possible. Any sequence is possible.

Commands without operands		
Command	**Function**	**Comment**
NOP	NO OPERATION	see note
EP	PROGRAM END	Initiates the I/O data replacement and begins a new cycle
HLT	HALT	Processing is halted and the outputs are set to 0
(	LEFT BRACKET	coreesponds to the “AND LEFT BRACKET” function [*(]
)	RIGHT BRACKET	
O(	EMPTY OR LEFT BRACKET	Double instruction. Corresponds to the “OR LEFT BRACKET” function [*(]
SC	SET CARRY	The CARRY is set to 1 (not dependent on RES)
RC	RESET CARRY	The CARRY is set to 0 (not dependent on RES)
)N	RIGHT BRACKET, NEGATION	Negation of the bracket contents

Note: The NOP command has no effect on the program seguence. Both commands are used to reserve memory space during program start-up in order to carry out changes in the program with the “Replace” ONLINE command in monitor operation.

1070 073 737-103 (98.12) GB

OPT / OPR	1.OPD. 2.OPD I / OPA	S Z / OID PAR	POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Jump commands																	
JP		[R]		●													Jump by the number of Opcode bytes indicated in the register
JP		SY		●													Unconditional jump
JPC		SY		●					1								Conditional jump with RES = 1
JPCI		SY		●					0								Inverted jump with RES = 0
JPZ		SY		●									1				Jump zero with Z = 1 (equality)
JPP		SY		●								0					Jump plus with N = 0 (arithmetic >=)
JPN		SY		●									0				Jump not zero with Z = 0 (inequality)
JPM		SY		●								1					Jump minus with N = 1 (arithmetic <)
JPO		SY		●						1							Jump overflow with O = 1
JPCY		SY		●							1						Jump carry with C = 1 (logic <)
JPLG		SY		●										1			Jump logic greater with LG = 1)
JPAG		SY		●											1		Jump arithmetic greater with AG = 1
JPCN		SY		●							0						Jump carry with C = 0 (logic >=)
JPCZ		SY		●										0			Jump with carry or Z = 1 (logic <=)
JPON		SY		●						0							Jump overflow not with O = 0
JPMZ		SY		●											0		Jump with negative (minus) or Z =1, i.e. arithmetic <=

1070 073 737-103 (98.12) GB

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module call commands																	
CM		PM	SY, PM	●													Unconditional module call
CMC		PM	SY, PM	●					1			1/0	0/1				Conditional module call with RES = 1 after succesful call N = 1 else Z = 1
CMCI		PM	SY, PM	●					0			0/1	1/0				Inverted module call with RES = 0 after succesful call Z = 1, else N = 1
CMZ		PM	SY, PM	●									1				Module call zero with Z = 1 (equality)
CMP		PM	SY, PM	●								0					Module call plus with N = 0 (arithmetic >=)
CMN		PM	SY, PM	●									0				Module call not zero with Z = 0 (inequality)
CMM		PM	SY, PM	●								1					Module call minus with N = 1 (arithmetic <)
CMO		PM	SY, PM	●						1							Module call overflow with O = 1
CMCY		PM	SY, PM	●							1						Module call carry with C = 1 (logic <)
CMLG		PM	SY, PM	●										1			Module call logic greater with LG = 1)
CMAG		PM	SY, PM	●											1		Module call arithmetic greater with AG = 1
CMCN		PM	SY, PM	●							0						Module call with C = 0 (logic > =)
CMCZ		PM	SY, PM	●										0			Module call with Carry or Z = 1 (logic <=)
CMON		PM	SY, PM	●						0							Module call overflow Not with O = 0
CMMZ		PM	SY, PM	●											0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module call commands																	
CM		DM	SY, DM	●													Unconditional module call
CMC		DM	SY, DM	●					1								Conditional module call with RES = 1
CMCI		DM	SY, DM	●					0								Inverted module call with RES = 0
CMZ		DM	SY, DM	●									1				Module call zero with Z = 1 (equality)
CMP		DM	SY, DM	●								0					Module call plus with N = 0 (arithmetic >=)
CMN		DM	SY, DM	●									0				Module call not zero with Z = 0 (inequality)
CMM		DM	SY, DM	●								1					Module call minus with N = 1 (arithmetic <)
CMO		DM	SY, DM	●						1							Module call overflow with O = 1
CMCY		DM	SY, DM	●							1						Module call carry with C = 1 (logic <)
CMLG		DM	SY, DM	●										1			Module call logic greater with LG = 1)
CMAG		DM	SY, DM	●											1		Module call arithmetic greater with AG = 1
CMCN		DM	SY, DM	●							0						Module call with C = 0 (logic > =)
CMCZ		DM	SY, DM	●										0			Module call with Carry or Z = 1 (logic < =)
CMON		DM	SY, DM	●						0							Module call overflow Not with O = 0
CMMZ		DM	SY, DM	●											0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=
CX CXC CXCI		DM	SY, DM	●													Call 2nd data module direct

OPT OPR	1.OPD. I OPA	2.OPD S Z OID PAR	POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module call commands																	
CM		PM	SY, PM	●													Unconditional module call (n = 0-62)
CMC		PM	SY, PM	●					1								Conditional module call with RES = 1
CMCI		PM	SY, PM	●					0								Inverted module call with RES = 0
CMZ		PM	SY, PM	●									1				Module call zero with Z = 1 (equality)
CMP		PM	SY, PM	●								0					Module call plus with N = 0 (arithmetic >=)
CMN		PM	SY, PM	●									0				Module call not zero with Z = 0 (inequality)
CMM		PM	SY, PM	●								1					Module call minus with N = 1 (arithmetic <)
CMO		PM	SY, PM	●						1							Module call overflow with O = 1
CMCY		PM	SY, PM	●							1						Module call carry with C = 1 (logic <)
CMLG		PM	SY, PM	●										1			Module call logic greater with LG = 1)
CMAG		PM	SY, PM	●											1		Module call arithmetic greater with AG = 1
CMCN		PM	SY, PM	●							0						Module call with C = 0 (logic > =)
CMCZ		PM	SY, PM	●										0			Module call with Carry or Z = 1 (logic < =)
CMON		PM	SY, PM	●						0							Module call overflow not with O = 0
CMMZ		PM	SY, PM	●											0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=

OPT	1.OPD. 2.OPD		POSSIBLE OPERANDS	ADDR TYPE					STATUS								Commend
	I	S Z															
OPR	OPA	OID PAR		Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	O	C	N	Z	LG	AG		
Module call commands																	
CM		P	P														Unconditional module call
CMC		P	P						1								Conditional module call with RES = 1
CMCI		P	P						0								Inverted module call with RES = 0
CMZ		P	P										1				Module call zero with Z = 1 (equality)
CMP		P	P									0					Module call plus with N = 0 (arithmetic >=)
CMN		P	P										0				Module call not zero with Z = 0 (inequality)
CMM		P	P									1					Module call minus with N = 1 (arithmetic <)
CMO		P	P							1							Module call overflow with O = 1
CMCY		P	P								1						Module call carry with C = 1 (logic <)
CMLG		P	P											1			Module call logic greater with LG = 1)
CMAG		P	P												1		Module call arithmetic greater with AG = 1
CMCN		P	P								0						Module call with C = 0 (logic > =)
CMCZ		P	P											0			Module call with Carry or Z = 1 (logic < =)
CMON		P	P							0							Module call overflow Not with O = 0
CMMZ		P	P												0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=
CX CXC CXCI		for P=DM	for P=DM														Module call for 2nd data module

1070 073 737-103 (98.12) GB

OPT	1.OPD. 2.OPD I	S	Z	POSSIBLE OPERANDS	ADDR TYPE					STATUS								Comment
OPR	OPA	OID PAR			Absolute	Reg.	Direct	Param.	Reg.Ind.	RES	O	C	N	Z	LG	AG		
Module call commands																		
CM		P,n		P				●										Unconditional module call (n = 0-62)
CMC		P,n		P				●		1								Conditional module call with RES = 1
CMCI		P,n		P				●		0								Inverted module call with RES = 0
CMZ		P,n		P				●						1				Module call zero with Z = 1 (equality)
CMP		P,n		P				●					0					Module call plus with N = 0 (arithmetic >=)
CMN		P,n		P				●						0				Module call not zero with Z = 0 (inequality)
CMM		P,n		P				●					1					Module call minus with N =1 (arithmetic <)
CMO		P,n		P				●			1							Module call overflow with O = 1
CMYC		P,n		P				●				1						Module call carry with C = 1 (logic <)
CMLG		P,n		P				●							1			Module call logic greater with LG = 1)
CMAG		P,n		P				●								1		Module call arithmetic greater with AG = 1
CMCN		P,n		P				●				0						Module call with C = 0 (logic > =)
CMCZ		P,n		P				●							0			Module call with Carry or Z = 1 (logic < =)
CMON		P,n		P				●			0							Module call overflow not with O = 0
CMMZ		P,n		P				●								0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=

OPT / OPR	1.OPD. I OPA	2.OPD S OID PAR	Z / POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module call commands																	
CM		[A]	[R]					●									Unconditional module call
CMC		[A]	[R]					●	1								Conditional module call with RES = 1
CMCI		[A]	[R]					●	0								Inverted module call with RES = 0
CMZ		[A]	[R]					●					1				Module call zero with Z = 1 (equality)
CMP		[A]	[R]					●				0					Module call plus with N = 0 (arithmetic >=)
CMN		[A]	[R]					●					0				Module call not zero with Z = 0 (inequality)
CMM		[A]	[R]					●				1					Module call minus with N = 1 (arithmetic <)
CMO		[A]	[R]					●		1							Module call overflow with O = 1
CMCY		[A]	[R]					●			1						Module call carry with C = 1 (logic <)
CMLG		[A]	[R]					●						1			Module call logic greater with LG = 1)
CMAG		[A]	[R]					●							1		Module call arithmetic greater with AG = 1
CMCN		[A]	[R]					●			0						Module call with C = 0 (logic > =)
CMCZ		[A]	[R]					●						0			Module call with Carry or Z = 1 (logic < =)
CMON		[A]	[R]					●		0							Module call overflow Not with O = 0
CMMZ		[A]	[R]					●							0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=

Note: [R] only possible with the PM and DM operands as a prefix.

OPT OPR	1.OPD. 2.OPD I OPA	S OKN PAR	Z POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module call commands																	
CM		[A], n	[R]					●									Unconditional module call
CMC		[A], n	[R]					●	1								Conditional module call with RES = 1
CMCI		[A], n	[R]					●	0								Inverted module call with RES = 0
CMZ		[A], n	[R]					●					1				Module call zero with Z = 1 (equality)
CMP		[A], n	[R]					●				0					Module call plus with N = 0 (arithmetic >=)
CMN		[A], n	[R]					●					0				Module call not zero with Z = 0 (inequality)
CMM		[A], n	[R]					●				1					Module call minus with N = 1 (arithmetic <)
CMO		[A], n	[R]					●		1							Module call overflow with O = 1
CMCY		[A], n	[R]					●			1						Module call carry with C = 1 (logic <)
CMLG		[A], n	[R]					●						1			Module call logic greater with LG = 1)
CMAG		[A], n	[R]					●							1		Module call arithmetic greater with AG = 1
CMCN		[A], n	[R]					●			0						Module call with C = 0 (logic > =)
CMCZ		[A], n	[R]					●						0			Module call with Carry or Z = 1 (logic < =)
CMON		[A], n	[R]					●		0							Module call overflow Not with O = 0
CMMZ		[A], n	[R]					●							0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=

Note: [R] only possible with the PM operand as a prefix.

OPT OPR	1.OPD. 2.OPD I OPA	S OID PAR	Z POSSIBLE OPERANDS	ADDR TYPE Absolute	Reg.	Direct	Param.	Reg.Ind.	STATUS RES	O	C	N	Z	LG	AG		Comment
Module end commands																	
EM																	Unconditional module end
EMC									1								Conditional module end
EMZ													1				Module end zero
EMP												0					Module end plus
EMN													0				Module end not zero
EMM												1					Module end minus
EMO										1							Module end overflow
EMCY											1						Module end carry
EMLG														1			Module end logic greater
EMAG															1		Module end arithmetic greater
EMCN											0						Module end with C = 0 (logic > =)
EMCZ														0			Module end with carry or Z =1 (logic <=)
EMON										0							Module end overflow not
EMMZ															0		Module call with negative (minus) or Z = 1, i.e. arithmetic <=
EMCI									0								Module end with RES = 0

3.12 Command execution times

Bit commands (time in µs)

A	B	I0.0	0.23
AN	B	I0.0	0.23
O	B	I0.0	0.23
ON	B	I0.0	0.23
A	B	I[A]	0.9
A	B	D[A]	1
S	B	O0.0	0.30
S	B	D[A]	1.15
R	B	O0.0	0.30
=	B	O0.0	0.30
A	B P0		
	P0=I0.0		5.8
	P0=D0.0		6.7
AN, O, ON as A			
S	B	P0	
	P0=O0.0		6.1
	P0=D0.0		6.5
S	B	M[A]	0.9
R =	as S		
TST	B	A,n	0.295
TSTZ	B	A,n	0.295

Byte, word and DW commands (time in µs)

A	W	K0,A	0.365
A	W	P0,A	
	P0=I0		5.2
	P0=D0		6.5
	P0=K0		5.8
	P0=T0		5.95
	P0=II0		43.5
A	W	I0,A	0.4
A	W	D0,A	0.8
A	W	D[A],A	0.95
A	BY	II0,A	31.5
A	W	II0,A	37
A	DW	II0,A	50
A	W	B,A	0.4
A	W	M[B],A	0.73
O, XO, as A			
AN	W	K0,A	0.7
AN	W	P0,A	
	P0=I0		5.4
	P0=D0		6.7
	P0=K0		6.0
	P0=T0		6.1
	P0=II0		43.5
AN	W	I0,A	0.75
AN	W	D0,A	0.9
AN	W	D[A],A	1.3
AN	BY	II0,A	32
AN	W	II0,A	37
AN	DW	II0,A	50
AN	W	B,A	0.76
AN	W	M[B],A	1.44
ON, XON as AN			

Timer commands (time in µs)

```
SP          A,T0
RES   falling        4.7
RES   rising         5.4
RES   stable         2.7

SP          A,P0
      P0=T0
RES   falling        6.5
RES   rising         7.2
RES   stable         4.5

SP          A,T[B]
RES   falling        4.6
RES   rising         5.5
RES   stable         2.7

SPE, SR, SF, SRE as SP

RT          T0
RES=0                1.75
RES=1                3.5

RT          P0
      P0=T0
RES=0                3.5
RES=1                4.5

RT          T[A]
RES=0                1.8
RES=1                3.8
```

Counter commands (time in µs)

```
CU          C0
RES   falling        3.5
RES   rising         3.7
RES   stable         2.5

CU          P0
     P0=C0
RES   falling        6.5
RES   rising         6.8
RES   stable         5.8

CU          C[A]
RES   falling        2.6
RES   rising         2.7
RES   stable         2.3

CD as CU

SC          A,C0
RES   falling        3.6
RES   rising         4.4
RES   stable         2.4

SC          A,P0
      P0=T0
RES   falling        7.2
RES   rising         8.5
RES   stable         6.5

SC          A,C[B]
RES   falling        4
RES   rising         5.4
RES   stable         3.7

RC as SC
```

Comparison commands (time in µs)

CPL	W	A,B	2.7
CPLA	W	I0,A	0.32
CPLA	W	D0,A	0.4
CPLA	W	D[A]	0.8
CPLA	BY	II0,A	31.5
CPLA	W	II0,A	37
CPLA	DW	II0,A	50
CPLA	W	B,A	0.3
CPLA	W	M[B],A	0.6
CPLA	W	K0,A	0.26
CPLA	W	P0,A	
	P0=I0		5
	P0=D0		6.4
	P0=K0		5.7
	P0=T0		5.5
	P0=II0		43.5

Code conversions (time in µs)

BID	BY	A	3.95
BID	W	A	6.2
BID	DW	A	13
DEB	BY	A	4.6
DEB	W	A	10.5
DEB	DW	A	24

Exchange command (time in µs)

SWAP	W	A	0.95

Stack commands (time in µs)

PUSH	DW	A	2.25
POP	DW	A	2.25

Arithmetic commands (time in µs)

ADD	W	I0,A	0.42
ADD	W	D0,A	0.75
ADD	W	D[A]	0.92
ADD	BY	II0,A	31.5
ADD	W	II0,A	37
ADD	W	II0,A	50
ADD	W	B,A	0.45
ADD	W	M[B],A	0.9
ADD	W	K0,A	0.38
ADD	W	P0,A	
	P0=I0		5.3
	P0=D0		6.6
	P0=K0		5.9
	P0=T0		5.6
	P0=II0		43.5

SUB, ADC, SBB as ADD

MUL	BY	A,B	1.95
MUL	W	A,B	2.35
MUL	DW	A,B	3.7
MUL	BY	K10,A	1.95
MUL	W	K10,A	2.35
MUL	DW	K10,A	3.55
DIV	BY	A,B	2.2
DIV	W	A,B	3.3
DIV	DW	A,B	4.5
DIV	BY	K10,A	2.1
DIV	W	K10,A	3.2
DIV	DW	K10,A	4.3
INC	W	A,n	0.36
INC	W	A,[C]	0.43

DEC as INC

TC	W	A	0.23
N	W	A	0.3

Load commands (time in µs)

L	W	I0,A	0.3
L	W	D0,A	0.7
L	W	D[A],A	0.8
L	BY	II0,A	31.5
L	W	II0,A	37
L	DW	II0,A	50
L	W	B,A	0.31
L	W	M[B],A	0.9
L	W	K0,A	0.29
L	W	P0,A	
	P0=I0		5
	P0=D0		6.3
	P0=K0		5.6
	P0=T0		5.4
	P0=II0		43.5

Transfer commands (time in µs)

T	W	A,O0	0.36
T	W	A,D0	0.5
T	W	A,D[A]	0.95
T	BY	A,IO0	30
T	W	A,IO0	35
T	DW	A,IO0	46
T	W	B,A	0.32
T	W	A,M[B]	0.5
T	W	A,P0	
	P0=O0		5.3
	P0=D0		6

Shift commands (time in µs)

SLR	W	A,n	0.28
SLR	W	A,[C]	0.42

SLL, RCR, SAR, ROR, ROL
as SLR

Commands without operands (time in µs)

NOP		0.1
(		0.23
)		0.23
SCY		0.1
RCY		0.1
*	N	2.9

Branch instructions (time in µs)

JP	-label	0.32
JPC	-label	
	not performed	0.38
	performed	0.58

other branch instructions
as JP

Module call commands (time in µs)

CMC	PM0	
	not performed	0.6
	performed	12.5
CMC	DM0	
	not performed	0.6
	performed	3
CMC	PM0,n	
	not performed	0.65
	performed	13.5
EM		8.5
EMC		
	not performed	0.7
	performed	9.5

Bosch-Automationstechnik

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Industriehydraulik
Postfach 30 02 40
D-70442 Stuttgart
Telefax (07 11) 8 11-18 57

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Fahrzeughydraulik
Postfach 30 02 40
D-70442 Stuttgart
Telefax (07 11) 8 11-17 98

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Pneumatik
Postfach 30 02 40
D-70442 Stuttgart
Telefax (07 11) 8 11-89 17

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Montagetechnik
Postfach 30 02 07
D-70442 Stuttgart
Telefax (07 11) 8 11-77 77

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Telefax (0 60 62) 78-4 28

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Schraub- und Einpreßsysteme
Postfach 11 61
D-71534 Murrhardt
Telefax (0 71 92) 22-1 81

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Entgrattechnik
Postfach 30 02 07
D-70442 Stuttgart
Telefax (07 11) 8 11-34 75

Technische Änderungen vorbehalten

Ihr Ansprechpartner

BOSCH

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Telefax (0 60 62) 78-4 28

1070 073 737-103 (98.12) GB · HB NC· AT/VMS1 · Printed in Germany