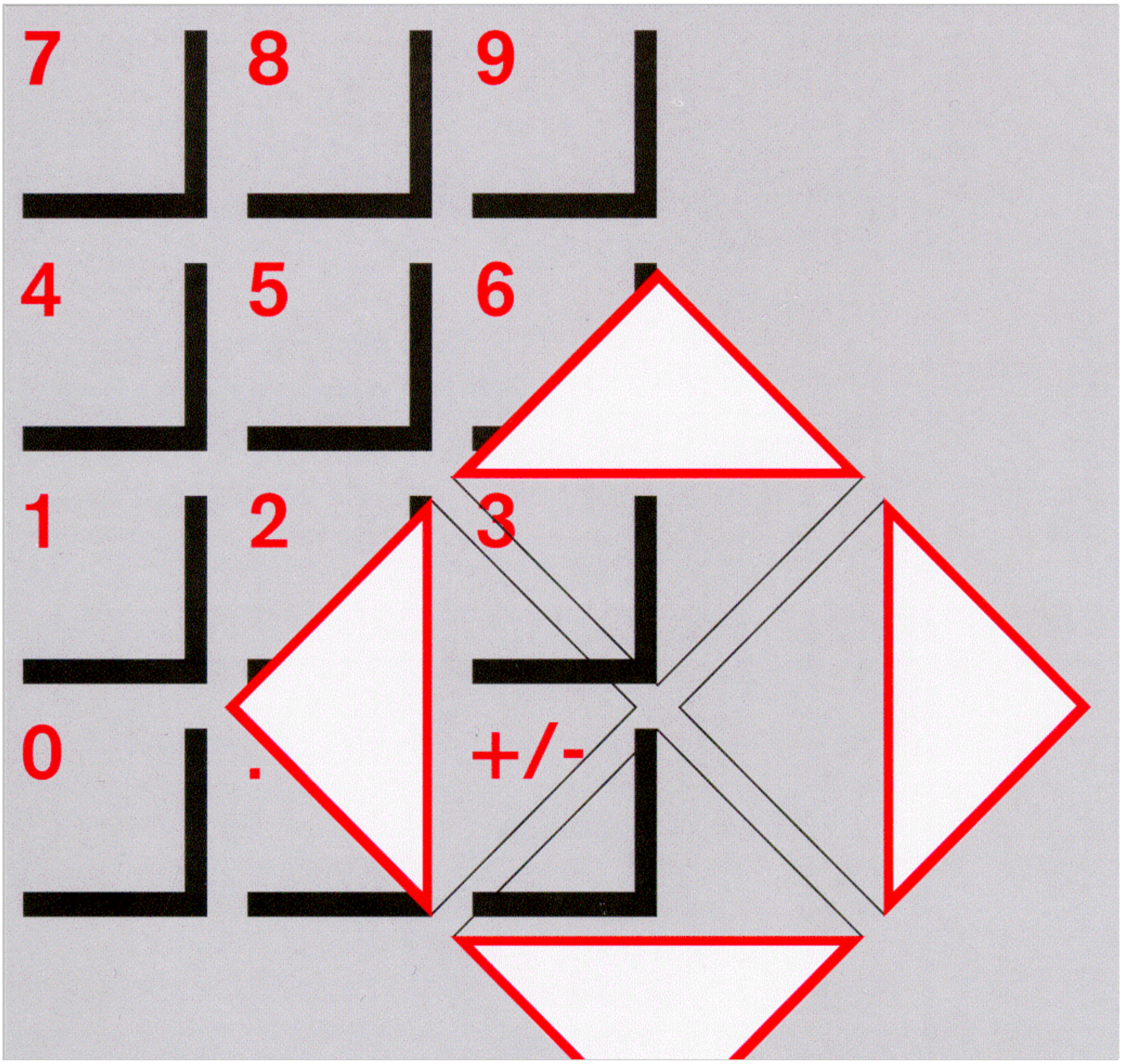
Typ3 osa / PNC

iPCL
System Description and Programming Manual



Edition 101

Rexroth
Bosch Group

Typ3 osa / PNC

iPCL
System Description and Programming Manual

1070 073 875-101 (02.04) GB

Software release: V7.x

Discretionary charge 12.– EUR

Contents

Notes:

1 Safety Instructions

Before you start working with the iPCL , we recommend that you thoroughly familiarize yourself with the contents of this manual. Keep this manual in a place where it is always accessible to all users.

1.1 Intended use

This manual contains information required for the proper use of this product. However, for reasons of structural clarity, the manual cannot provide exhaustive details regarding all available combinations of functional options. Similarly, it is feasible to consider every conceivable integration or operating scenario within the confines of this manual.

The Typ3 osa and PNC controls serve as

- 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 hereunder

- 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, i.e.
  - no direct connection to public low-voltage power supply,
  - connection to the medium- or high-voltage system via a transformer.

  In residential environments, in trade and commerce as well as small enterprises class A equipment may only be used if the following warning is attached:

☞ **This is a Class A device. In a residential area, this device may cause radio interference. In such case, the user may be required to introduce suitable countermeasures, and to bear the cost of the same.**

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

1.2 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.

This manual is intended for **project engineers and NC specialists**, who are familiar with programmable logic controllers (PLC). A special knowledge of how to configure and commission electrical equipment is also required

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 may result in serious bodily injury or material damage.

Only electrotechnicians as recognized under IEV 826-09-01 (modified) who are familiar with the contents of this manual may install and service the products described.

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.

With regard to the foregoing, please note our comprehensive range of training courses. For current information, the web shop and online booking of seminars please visit our website http://www.bosch.de/at/didactic. Our training center will be pleased to provide you with further information, telephone: (+49) (0 60 62) 78–258.

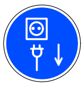
1.3 Safety markings on products

Warning of dangerous electrical voltage!

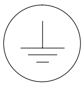
DANGER! Corrosive battery acid!

Electrostatically sensitive components!

Hazardous light emissions
(optical fibre cable emitters)!



Disconnect mains power before opening!



Lug for connecting PE conductor only!

Connection of shield conductor only

1.4 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 injury**.

DANGER
This symbol is used wherever insufficient or lacking compliance with instructions may result in **personal injury**.

CAUTION
This symbol is used wherever insufficient or lacking compliance with instructions may result in **damage to equipment or data files**.

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

★ This symbol is used if user activities are required.

1.5 Safety instructions for the described product

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
Risk of personal injury and equipment damage!
Always subject new programmes to initial tests while inhibiting axis movements. For this purpose, as a function of the AUTOMATIC mode, the controller provides the option to block axis movements or auxiliary functions by means of special softkey commands.

DANGER
Incorrect or undesired control unit response!
Bosch accepts no liability for damage resulting from the execution of an NC program, an individual NC block or the manual movement of axes!
Furthermore, Bosch accepts no liability for consequential damage which could have been avoided by programming the PLC appropriately!

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

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

Measuring or test activities on the live system are reserved to qualified electrical personnel!

DANGER
Tool or axis movements!
Feed and spindle motors generate very powerful mechanical forces and can accelerate very quickly due to their high dynamics.

- **Always stay outside the danger area of an active machine tool!**
- **Never deactivate safety-relevant functions!**
- **Report any malfunction of the unit to your servicing and repairs department immediately!**

CAUTION
Use only spare parts approved by Bosch!

CAUTION
Danger to the module!
All ESD protection measures must be observed when using the module! Prevent electrostatic discharges!

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

- Personnel responsible for storage, transport, and handling must have training in ESD protection.
- ESD-sensitive components must be stored and transported in the 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 may come into 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 1 MΩ resistor.
- ESD-sensitive components may by no means come into 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.

1.6 Documentation, software release and trademarks

Documentation

This manual provides details of the programming and operation of the iPCL. Not included are general procedures for project management and installation of controllers and their associated hardware.

Overview of available documentation	**Part no.**	
	German	**English**
Typ3 osa – Interface conditions for project engineering and maintenance	1070 073 704	1070 073 736
Typ3 osa – Software installation	1070 073 796	1070 073 797
Decription of functions	1070 073 870	–
MACODA Operation and configuration of the machine parameters	1070 073 705	1070 073 742
Operating instructions Standard operator interface	1070 073 726	1070 073 739
Operating instructions – Diagnostics Tools	1070 073 779	1070 073 780
Error Messages	1070 073 798	1070 073 799
PLC project planning manual, Software interfaces of the integrated PLC	1070 073 728	1070 073 741
iPCL system description and programming manual	1070 073 874	1070 073 875
ICL700 system description, Program structure of the integrated PLC ICL700	1070 073 706	1070 073 737
DIN programming manual for programming to DIN 66025	1070 073 725	1070 073 738
CPL programming manual	1070 073 727	1070 073 740
CPL-Debugger Operating instructions	1070 073 872	–
Tool Management – Parameterization	1070 073 782	1070 073 793
Software PLC Development environment for Windows NT	1070 073 783	1070 073 792
Measuring cycles for touch-trigger switching probes	1070 073 788	1070 073 789
Universal Milling Cycles	–	1070 073 795

☞ **In this manual the floppy disk drive always uses drive letter A:, and the hard disk drive always uses drive letter C:.**

Special keys or key combinations are shown enclosed in pointed brackets:

- Named keys: e.g., <Enter>, <PgUp>, <Del>
- Key combinations (pressed simultaneously): e.g., <Ctrl> + <PgUp>

Release

☞ **The descriptive information contained in this manual applies to:**
Software version: V7.x

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 may be displayed as follows:

1. Click the right mouse button on the **My Computer** icon on your desktop.
2. Select **Properties**.

Trademarks

All trademarks of software installed on Bosch products upon delivery are the property of the respective manufacturer.

Upon delivery, all installed software is copyright-protected. The software may only be reproduced with the approval of Bosch or in accordance with the license agreement of the respective manufacturer.

MS-DOS® and Windows™ are registered trademarks of Microsoft Corporation.

PROFIBUS® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (user organization).

SERCOS interface™ is a registered trademark of the SERCOS interface Joint VDW/ZVEI Working Committee.

2 System Overview

2.1 Functionality

iPCL is a software PLC integrated into the NC control. Without additional hardware iPCL is integrated into:

- the PNC plug-in card
- the type 3 osa component group osa master P-L and osa master P-XL.

Thus a secure functionality is assured independently of Windows.

I/O's are connected via PROFIBUS-DP, enabling RM65M-16DP, B~IO-modules to be used, for example.

For the operation and programming of iPCL, the following configuration software is required:

WinSPS: Creation of the PLC application program with functional extensions for communication between PLC and NC (APS modules)

WinDP: Configuration of the PROFIBUS-DP

Communications with the WinSPS, WinDP and other programs are handled by the TCP/IP standard protocol with the use of the BUEP (Bosch transfer protocol) command language.

For the creation of the PLC program or individual program modules in the programming language C, the C compiler and linker are also required.

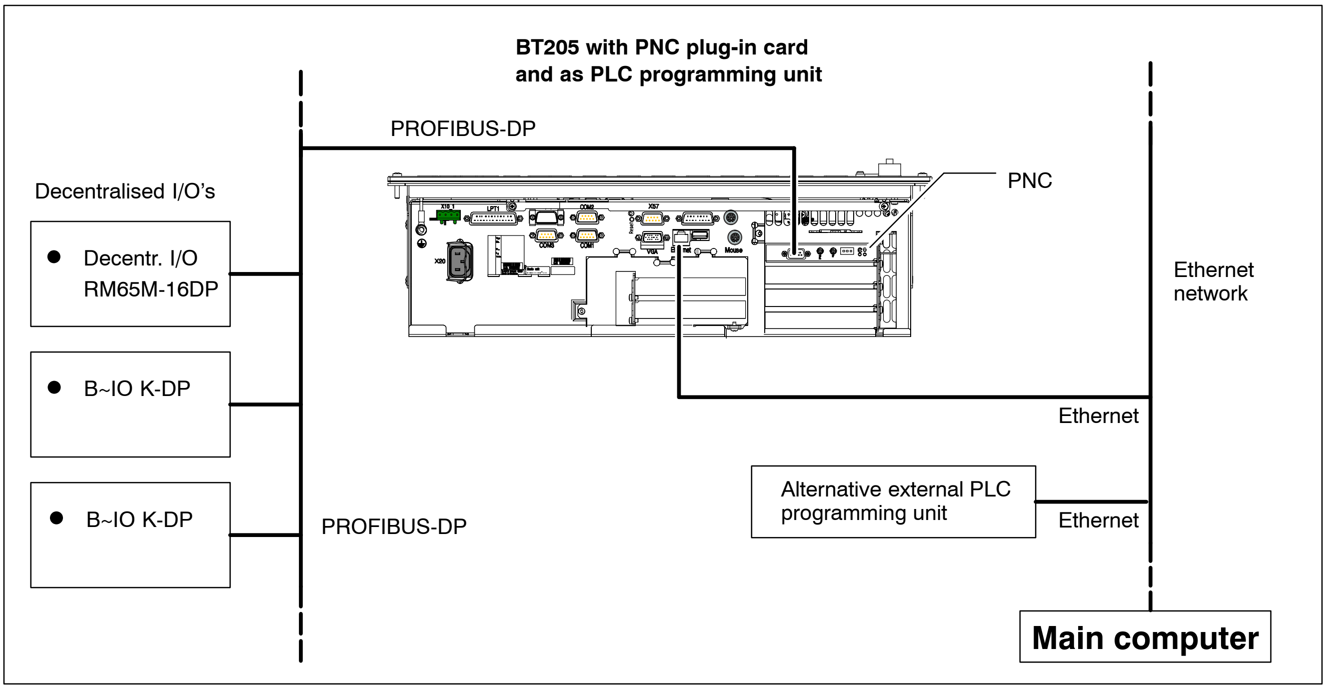
☞ **For additional essentials related to iPCL and to operating decentralized peripherals via the PROFIBUS-DP, refer to the Online Help in WinSPS and WinDP.**

2.2 Hardware platforms

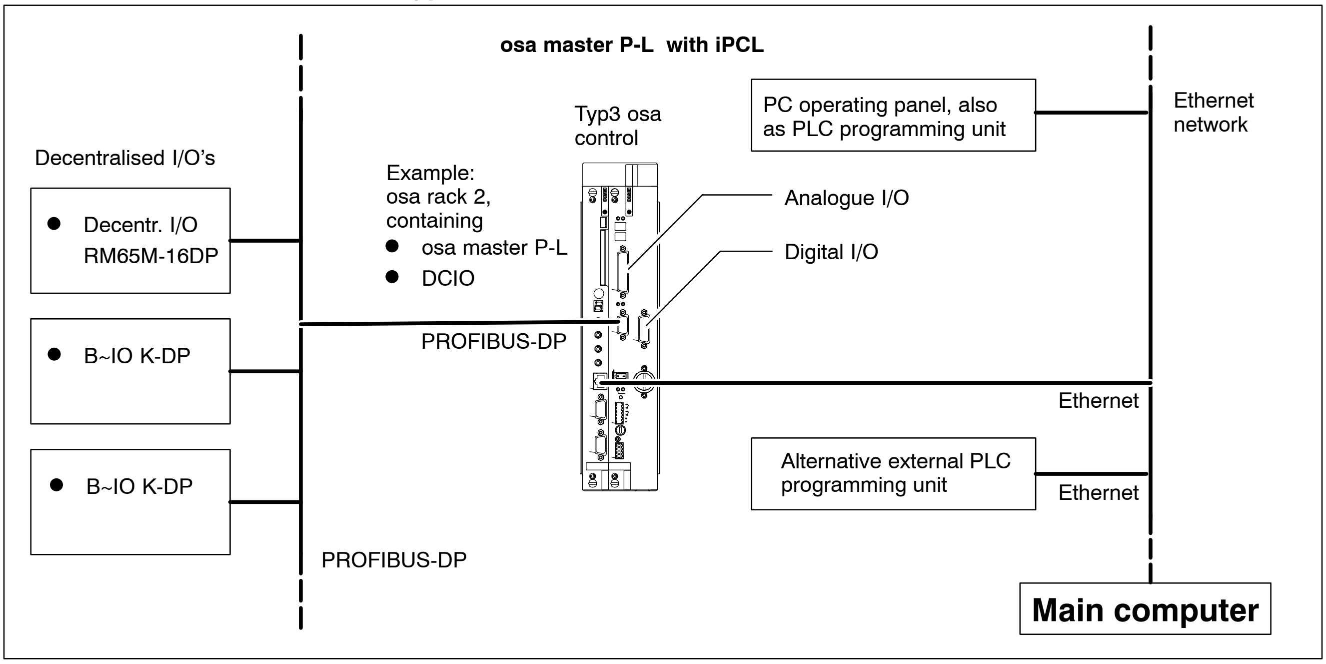
iPCL is integrated in:

- the PNC plug-in card
- the Typ3 osa component group osa master P-L and osa master P-XL.

iPCL in PNC (e.g. in the operating terminal BT205)



iPCL in the osa master P-L/XL (Typ3 osa)



2.3 iPCL extensions

The maximum I/O area and the PLC user memory (MACODA parameter 2060 00210) are determined by licence:

Type	Peripherals	User memory
iPCL_1 (PNC)	16 kb for inputs 16 kb for outputs	32 kbytes
iPCL_2 (PNC)	256 kb for inputs 256 kb for outputs	128 kbytes
iPCL_3 (PNC)	8 kb for inputs 8 kb for outputs	512 kbytes
iPCL_4 (osa master P-L/XL)	-256 bytes for inputs -256 bytes for outputs	Default: 200 kb depending on free memory in the osa master P-L/XL

Because the data field and data buffer are included in every hardware expansion level, they do not reduce the size of the user memory! Just like the program and organization modules, the data modules are stored in the PLC user memory.

Additional options

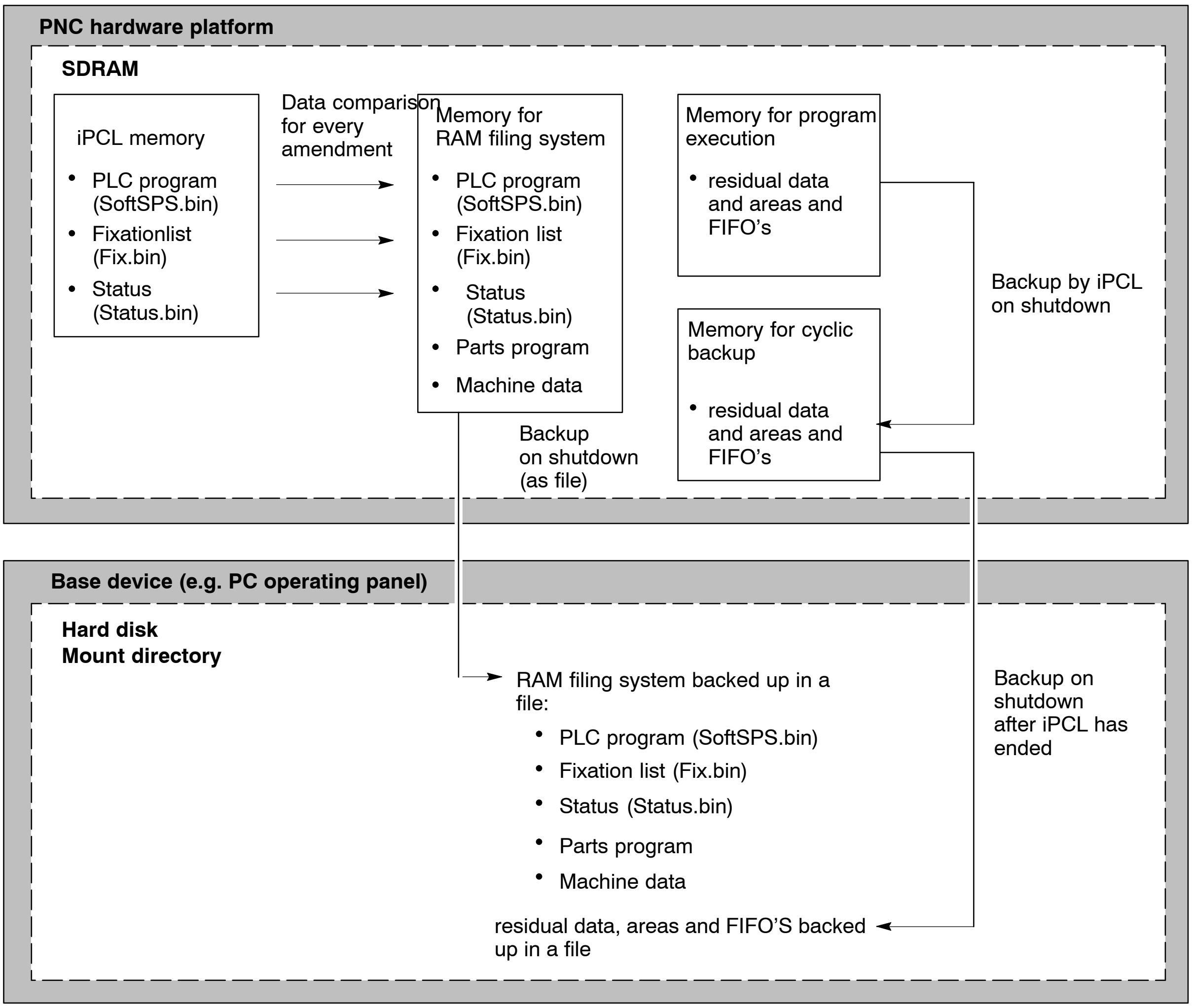
OPC server functions are available. It enables MADAP STUDIO to be used together with the PNC.

2.4 Data backup

PNC

iPCL uses PNC's own memory (SDRAM) and the hard disk of the base unit into which the PNC is plugged.

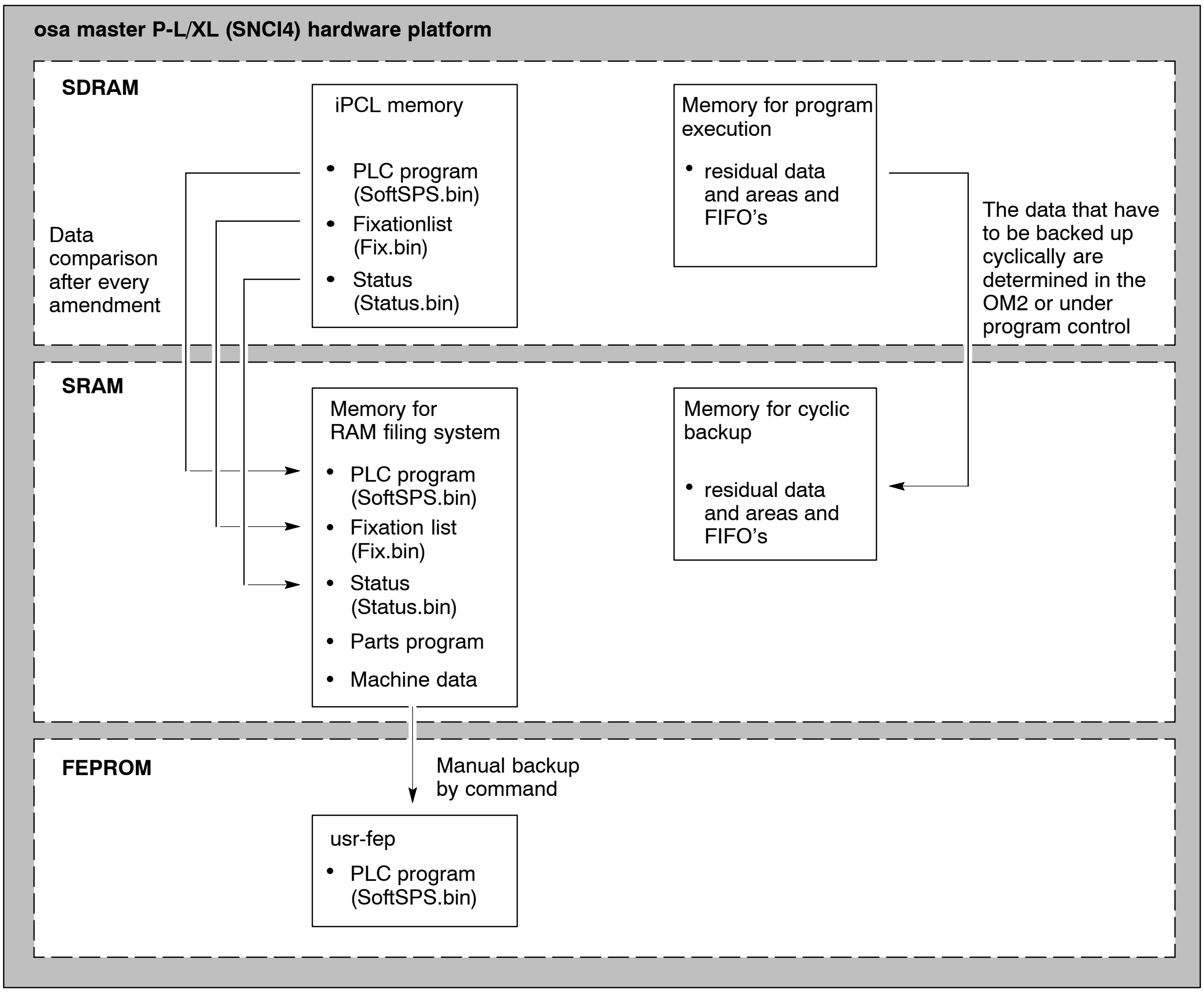
Optimum functional security of the iPCL in the PNC is attained by using a UPS (uninterruptible power supply), which bridges a potential power loss to allow essential PLC and NC data to be backed up to hard disk and leads to a delayed shut down of the Windows NT operating system.



osa master P-L/XL (SNCI4)

iPCL uses various memory areas of the component group osa master P-L/XL:

- SDRAM (dynamic mamory) for PLC program and data in use.
- SRAM (static memory) for PLC program and data for switched off control and cyclic backups.
- FEPROM for additional back up of the PLC program.



Notes:

3 Configuration

3.1 Connecting to the system

Registering iPCL via MACODA

iPCL has to be registered in the MACODA parameter 2060 00200. Apart from that further parameters can be changed:

- 2060 00200: Selection of the PCL. Must be set to iPCL (= 4).
- 2060 00210: Maximum size of the user program. For the PNC the size may be limited by licence, (for detailed information, refer to Section 2.3)
- 2060 00211: Max PLC computing time in %. (see Section 8).

Interfacing with Peripherals

The interfacing with peripherals is viy the serial filed bus system **PROFIBUS-DP** via the PROFIBUS-DP Busmaster interface:

- for **base devices with PNC**: on the ”PNC-PCI card”
- for the **Typ3 osa hardware**: on the component group ”osa dc I/O”

The maximum I/O area is determined by licence.

Reference list:

The bus master monitors the existence of slaves and transfers this data to the iPCL.

Error functions:

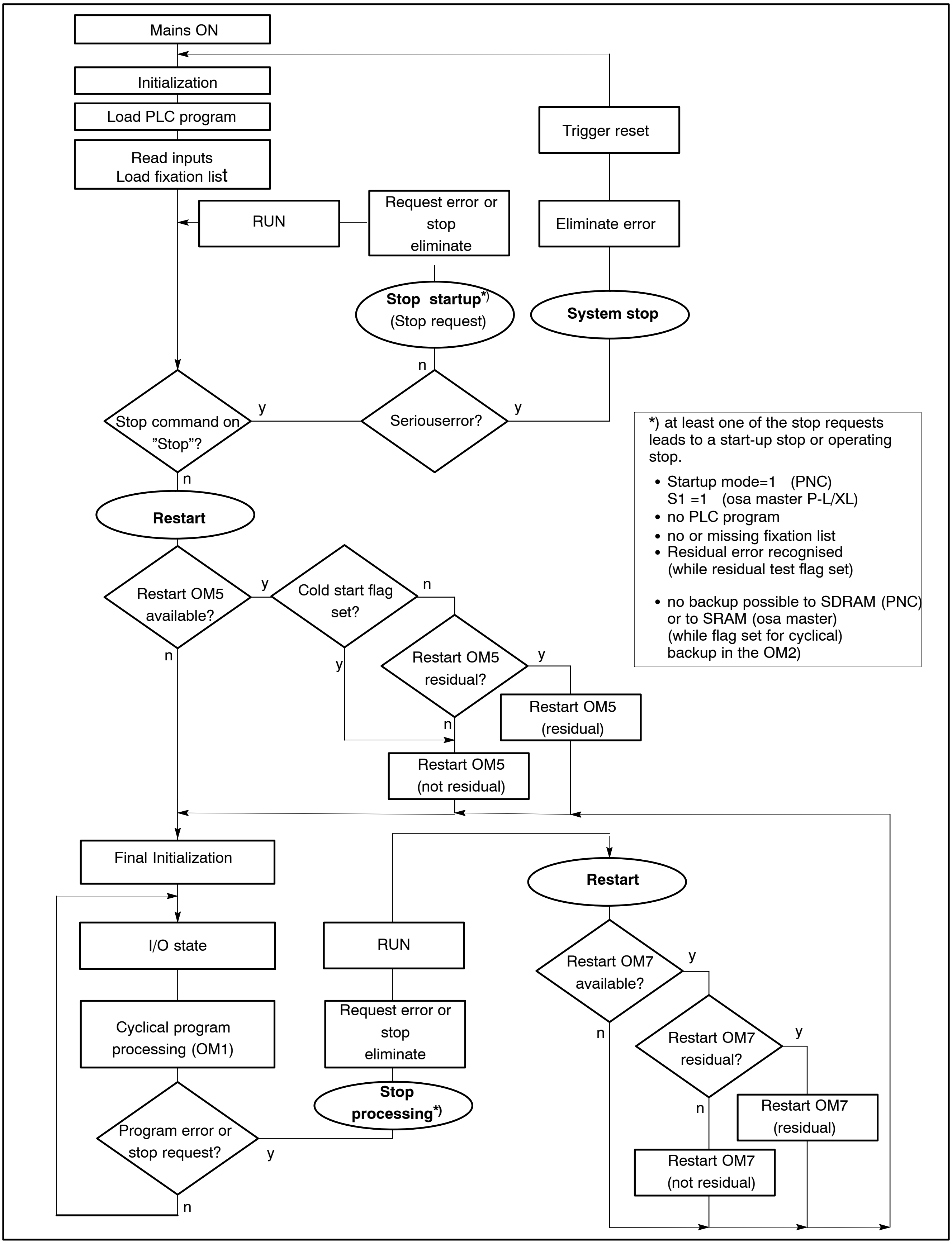
The error functions are dependent on the bus system used. The PROFIBUS-DP field bus features a comprehensive diagnostic system whose messages are made available by the iPCL bus master.

System clock management

The system timing, which can be processed in the PLC program via the system area, is generated by the clock source onboard the PC.

3.2 Startup of the iPCL

There follows the initialization and the start-up diagram for the iPCL. This procedure is the same for all hardware platforms.



3.2.1 Initialization of the iPCL

In the initialization phase the iPCL operating system starts up.

Initializing special markers

The special markers SM21.0 to SM31.7 (see below: “Exceptions”) are pre-initialized during “New start” and “Restart”. They are subsequently modified in accordance with their function.

Initialization values

SM 26	= $FFFF_H$
SM 31.1	= 1
All others	= 0

Exceptions for initial start

SM 20.0	Reset impulse for new start and restart Is set to 1 for iPCL new start and restart. Marker is deleted if OM1 has been processed at least once.
SM 20.1	Buffer fault Is set if the buffering of residual data was not correct.
SM 20.2	Flashing marker Flashes at 2 Hz after iPCL startup
SM 20.3	Block outputs Is set according to the requests for output blocking. Always updated during I/O state.
SM 20.4	Fixation markers Is set in accordance with the fixation request. Always updated during I/O state.
SM 20.5	Data backup error Is set if the buffering of residual data was not correct.
SM 20.6	Non-residual cold start Is set when the cold start has occurred and all residual areas were deleted.
SM 20.7	Reset impulse for new start and program load Is set to 1 for iPCL new start and after program load. Marker is deleted if OM1 has been processed at least once.

3.2.2 Startup diagram

Startup conditions

After initialization the actual iPCL **Startup** begins.
During startup an attempt is made by the PLC program to load various files and data. Here different settings and potential events during the run up are taken into account.

- Switch setting S1
- residual or non-residual startup characteristics (see page 3–5)
- after startup stop (for new start, see page 3–7)
- after processing stop (for restart, see page 3–8)

Switch setting S1

The loading of the PLC program and the data occurs with the pre-setting of the rotary switch S1 on the osa master P-L/XL or startup mode in "PNC control" for the PNC.

The following switch settings are possible:

Switchsetting or startup mode	**Characteristics** (PNC / osa master P-L/XL)
0	Startup from the RAM filing system. The PLC program and residual data are loaded. The PLC program is starting.
1	As 0, but the PLC program will **not** start.
2	The PLC program backed up on hard disk (PNC) or in the user FEPROM (usr-fep) and residual data are being loaded. The PLC program is starting.
4, 5	As 0.
6	As 2. Additionally the RAM filing system is being newly created. The fixation list is being deleted.
7	As 0. Simultaneously the backup of the RAM filing system is deleted.

Startup characteristics

The startup characteristics are dependent on whether events such as “stop” or “new start” have already occurred that cause a certain startup procedure:

- After a startup stop or an **initial** “Power ON” a “**New start**” is initiated for the iPCL.
- After a processing stop a “**Restart**” is initiated (see page 3–8).

☞ **Both startup types can occur non-residual or residual.**

For each startup type there is an organization module available, which if it has been programmed in the PLC program, will run depending on the stop condition that has occurred:

OM5: New start OM, non-residual or partially residual
OM7: Restart OM, non-residual or partially residual

If the startup OMs are not programmed in the PLC program, then the corresponding startup proceeds without OM processing. In all startup types the factors from the OM2 are used or if they are not available, then default values are used.

☞ **If iPCL starts up with default settings, then it is always a “non-residual startup”.**

The data affecting the system area (times for time-controlled OMs, residual limits) can then be modified in the respective startup OM.

☞ **For iPCL there is no full-residual restart but only a partial-residual startup. The areas of the markers, times and counters defined as residual are kept. A full-residual startup, where the PLC program continues at the exact place in the program where it was stopped, is not possible because also the NC, which is connected to the iPCL, does not recognize a residual startup.**
Consequently in the following text the expression “residual” in association with the iPCL is always to be interpreted as “partial-residual”!

Startup sequence

The iPCL startup sequence proceeds as residual or non-residual.

1. Load inputs
2. Overlay fixation: it already works for direct access from the startup OM. However, the output to the peripherals does not occur directly, and the output image is not updated.
3. Stop command query:
   - if stop then a system stop is carried out
   - if no stop then the new start OM(OM5) is processed. This OM permits the use of all PLC instructions (also applies to restart OM(OM7)), e.g., to set outputs, to initialize or start timer or counter values, to manipulate values in the system area (to influence initialization values), or to modify residual limits.
4. Final initialization: Once the startup OM has been processed, the final initialization is executed, utilizing the values from the system table and system area. Values such as time monitoring, OM time values, etc., are adopted or updated.Provided the respective setting has been made in the OM2, the specified data module is copied into the data buffer.
5. Execute complete I/O state

6. Start program processing on the OM1.
7. Start timeframe processing for the times and release time OM processing.

☞ **If for an osa master P-L/XL card the PLC program is switched again to STOP after the startup of the controller (also applies to loading with control STOP), the READY signal drops and only returns if the rotary switch is briefly turned to a setting > 7. This behaviour is determined by the hardware and cannot be influenced by software. When a program or module is reloaded without control stop, the READY signal stays on.**
In the PNC the READY signal returns automatically after the restart of the PLC program.

Cold start flag

When the cold start flag is set, this forces a non-residual startup. This flag can be manipulated by either operating system or PG.

- Operating system: When the iPCL is switched on the PLC program is loaded from the softsps.bin file into memory. If an error occurs in the course of this process, the cold start flag will be set.
- Programming unit (WinSPS): The cold start flag is set when ’loading entire program with reset of residual operands’.

3.2.3 Startup conditions

Startup without error

A startup without error occurs when, subsequent to error-free program processing, the controller (NC and iPCL) is cycled OFF and ON again:

- The PLC program can be loaded again error-free into the PLC user memory.
- The residual areas are error-free
- The selected new start OM(OM5) is processed.
- During creation by WinSPS, the fixation lists are immediately backed up in the filing system. These fixation lists are loaded when the controller is started up.
- The cold start flag is not set.
- The startup is executed, and cyclical program processing is started.
- After PLC has been put into STOP state by the programming unit, it does not remain in this state when PLC is restarted.

If an error occurs at this juncture, the iPCL will enter Process-STOP, and will no longer enter Startup-STOP.

Startup with startup STOP

In the event that during the startup subsequent to power ON a “hardware fault” or “STOP request” occurs, the iPCL will remain in **Startup-STOP** mode.

Reasons for a stop request can be:

- No PLC user program in the directory (PNC) or in the filing system (osa master P-L/XL).
- Startup mode (PNC) or
  osa master P-L/XL rotary switch set S1= 1 (corresponds to stop)
- Severe faults occurred during controller run-up, e.g. faults originating in the installation of peripheral drivers, initialization of PLC operating system, or communication channel setup.
  These faults produce the message ”System stop” and do not permit a retriggering, i.e. the controller has to be resarted (reset).
  Faults that produce a controller stop are reported to the NC and are displayed on the operating panel in the INFO dialogue of the standard BOF.
- Incorrect or non-existent fixation lists file in the PNC root directory. The iPCL stays in startup stop mode until a fixation list is loaded. Loading an ”empty” list deletes the fixation identifier.
- Flag for residual test has been set in OM2, and residual error has been detected.
- Flag for cyclical backup set in the OM2 and operand backup to the static RAM (osa master P-L/XL) or SDRAM (PNC) is not possible.

Startup stop mode is left as soon as the fault has been fixed. By command from the programming unit (WinSPS) or in the PNC control, startup stop mode can be left by switching to RUN.

☞ **After startup stop there always follows a “New start”.**

Startup with processing STOP

Once the program processing has begun with the OM1, and an error or a STOP request occurs, this will cause a **Process-STOP** condition.
This stop condition is left as soon as the fault has been fixed or if the reason for the stop has disappeared and the switch in the iPCL control panel was set to RUN.

The stop state can also be left by command from the programming unit (WinSPS).

Non-residual new start or restart

The **non-residual** startup mode is used in the following cases:

- The system area flag is set in OM2.
- Subsequent to a memory error, as this precludes a residual startup.
- A non-residual startup was requested from the PG (WinSPS) (only possible when loading).

To describe the process in detail:

- All image areas (residual and non-residual) are deleted.
- Fixation is deleted upon new start, and retained upon restart.
- Stored interrupts are deleted.
- Application stack is reset.
- Outputs are enabled.
- Inputs are loaded.

Residual new start or restart

The **residual** startup mode is used in the following cases:

- No memory error has occurred.
- and no non-residual startup was requested by PG (WinSPS) or via the OM2.

To describe the process in detail:

- Non-residual areas are deleted.
- Timer values are transferred.
- Outputs are enabled.
- Inputs are loaded.

☞ **If an error occurs while loading the residual data, an error message is generated and the PLC program does not start automatically. The residual data area is re-initialized. The PLC program can be started either with a renewed run up of the controller or with the PG (WinSPS).**

3.3 Data backup and residual characteristics of the iPCL

Data backup is essential so that relevant PLC data are available for continued processing even after a power failure when the RAM filing system was newly created or in an error situation.

Data

The following data and files are associated with a current project:

- Residual areas:
  - Identified data modules
  - Data fields and data buffers
  - Markers
  - Times
  - Counters
  - FIFOs
- Files:
  - PLC program
  - Fixation lists
  - Status

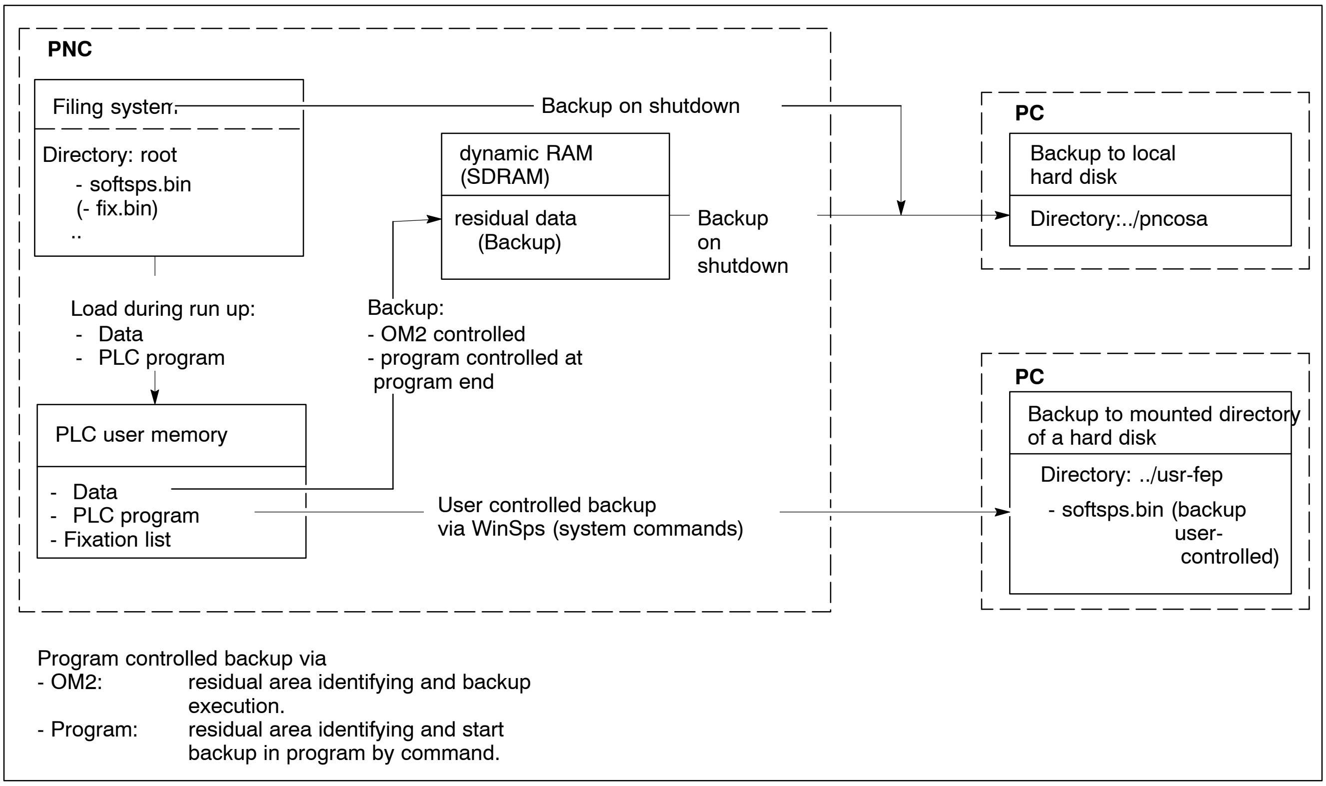
3.3.1 Data backup depending on hardware platform

PNC

PNC has a dynamic RAM (SDRAM) that does **not** allow storage of data when the controller is switched off.

The PLC application program is in the filing system, the data in the RAM. When the PNC is run up the data and the PLC program are automatically loaded from the local hard disk or from an additionally mounted directory of a hard disk (e.g. from a network computer) into the PLC user memory.

During program execution the user can, under program control, store certain residual data (cyclical) in a reserved SDRAM area, but the data will be lost if the controller is switched off.



☞ **During normal operations the user has available the operating functions, loading and storage of the PLC program and possibly backup of the PLC project in the usr-fep directory. Cyclical backups are controlled in the OM2 or via program.**

Power failure:
The PNC card is in a PC that must be attached to an “uninterruptible power supply” (UPS). The UPS ensures that in case of a power failure there is sufficient time for an orderly shutdown of iPCL and the Windows 95 or NT operating system. The essential backup of residual data, the PLC program, status and fixation list can all take place.

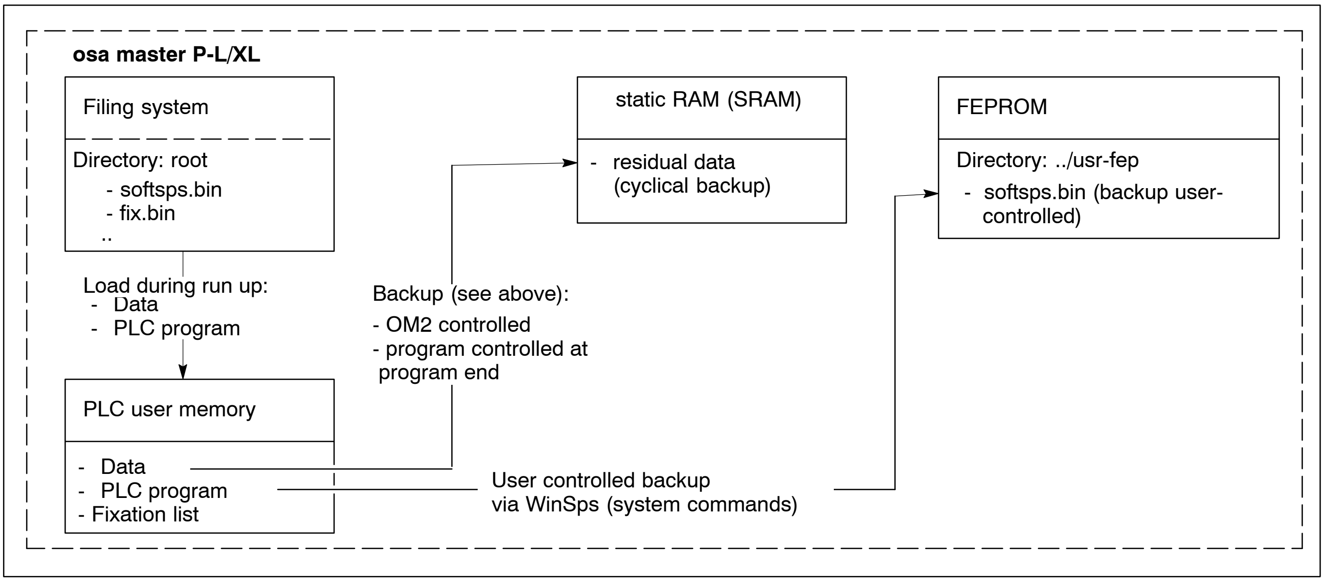
On the next startup PLC will load the entire backup data: the softsps.bin file, with respect to the PLC program, some other files and additional residual data that were backed up previously.

☞ **A backup of data to hard disk is not possible if Windows goes down but iPCL continues to run. The result is a loss of data as a re-boot is necessary.**

osa master P-L/XL

The “static RAM” (SRAM) of the osa master P-L/XL can save the state and values of residual data, areas, fixations, the PLC program, etc. even if the controller is **switched off**.

During operations the PLC application program is a file in the filing system of the Typ3 osa. From there it is loaded into the PLC application memory when the iPCL is run up.
PLC user data are loaded from the static RAM of the osa master P-L/XL into the PLC application memory.



Certain residual data designated by the user are saved under program control in a special RAM area (SRAM). After the controller is switched off these data and the filing system remain in tact. On the next run up these cyclically saved data and the PLC program are available.

Additionally the PLC program can be saved in the FEPROM (usr-fep), so that in case of a fault or if required the PLC can be loaded with the PLC program saved there.

☞ **During normal operations the user has available the operating functions, loading and storage of the PLC program and possibly backup of the PLC project in the usr-fep directory. Cyclical backups are controlled in the OM2 or via program.**

Power failure:In case of a power failure all relevant data from the last cyclical backup, the PLC program, fixation lists, etc. are in the SRAM. On the next run up PLC loads the PLC program saved in the SRAM and also takes into account the residual data from cyclical backups in the SRAM.

Optimal functional data security for iPCL is attained with the help of the static memory on the **osa master P-L/XL** . Every amendment is written to static memory, the contents of which are maintained even when the controller is switched off.

Selection of the residual data for cyclical backup

One can specify **which** residual data are to be backed up cyclically to the SRAM Typ3 osa or SDRAM in the PNC.

- The cyclical backup of residual areas is carried out during each I/O state. The establishment that data are to be backed up cyclically into static RAM or the osa master P-L/XL or RDRAM of the PNC occurs via **entries in the OM2** (see Section 3.3.2 Establishment of the residual areas in the OM2). At program end all residual data specified in the OM2 and identified residual areas for M / T / Z / DP / DF are defined.

  Note osa master P-L/XL: Because the process of writing to static RAM is much slower than that of writing to dynamic RAM, and thus causes the PLC cycle to be extended, the residual areas to be selected must be kept as small as possible.

- The cyclical backup procedure may be replaced by PLC instructions that perform the backup of the specified residual areas **upon request** (refer to Section 7.30 Backing up/loading residual areas): in the PLC program backup routines can be defined and executed at specific coordination points. This allows defined data to be backed up at specific times.
  For the backup of residual areas and of the modules designated as residual DM to the SRAM of the osa master P-L/XL or the PNC SDRAM the following measures are required:
  The residual identification E for the maximum 128 data modules must be specified in the symbol file (For example: DM1,E DM_K01).The residual limit definitions must also be declared in this case because, when copying from static into dynamic RAM, these areas must be known already during the startup sequence. The user disables the definition ”Cyclical backup”.

3.3.2 Defining residual areas in the OM2

```
;DW 2:  Initialization flag (entries permitted)
;--------------------------------------------------------------------------
;     Entry 0 = D O  N O T  test or execute function
;     Entry 1 = Verify and/or execute function

DEFW  W  2#0000000100000100
;       *******|*****||*       *: not used
;              |      |+------- Check nominal cycle time
;              |     +-------- Residual start if possible
;             +-------------- Copy data module into data buffer

;DW 3:  System settings (entries permitted)
;--------------------------------------------------------------------------
;     Entry 0 = D O  N O T  test or execute function
;     Entry 1 = Verify and/or execute function
;
DEFW  W  2#0000000001000000
;       *********||||||       *: not used
;                |||||+------ Markers        \  residual areas for
;                ||||+------- Times           \ cyclical Backup to the
;                |||+-------- Counters         \static RAM, as per
;                ||+--------- Data field       /per defined residual
;                |+---------- Data buffer     / limits
;                +----------- Data modules   /
;               +------------ cyclical Backup of the marked areas
```

```
;DW 7:  Number of first residual time (entries permitted)
;-----------------------------------------------------------------------
;     Entries from 0 to 256 are possible
;     128 = Residual  for timer loops T128 through T255
;     256 = No residual

DEFW W  128

;DW 8:  Number of first residual counter (entries permitted)
;-----------------------------------------------------------------------
;     Entries from 0 to 256 are possible
;     128 = Residual for counters Z128 to Z255
;     256 = No residual

DEFW W  128

;DW 9:  Number of first residual marker (entries permitted)
;-----------------------------------------------------------------------
;     Entries from 0 to 8192 are possible
;     128 = Residual from marker byte M128/marker bit M128.0; the residual
;           definition of residual limit via byte addresses
;     8192= No residual

DEFW W  4096

;DW 10: First residual address in data buffer (entries permitted)
;-----------------------------------------------------------------------
;     Entries from 0 to 512 are possible
;     256 = Residual from data buffer byte DP256
;     512 = No residual

DEFW W  256


;DW 33: First residual address in data field for backup to;        static
RAM (entries permitted)
;-----------------------------------------------------------------------
;     Entries of 0 and 32768 possible
;     16384 = Residual from data field byte DF16384 in static RAM
;     32768 = No residual in static RAM
;     Limit applies only to backup into static RAM; this area
;     takes precedence over the data field, the remainder of which is com-
pletely
;     backed up to hard disk for residual storage;

DEFW W  16384
```

3.3.3 Residual characteristics depending on hardware platform

Summary of residual characteristics under different hardware platforms

PNC with UPS	osa master P-L/XL
Backup of residual data is essential when powering off and on shutdown.	Backup of residual data into static RAM is required under program control for power off. I.e. power off immediately interrupts program processing and means that the data and residual conditions produced in **this** cycle cannot be backed up any more.
All residual areas/ranges can be backed up, provided they have been defined as residual. In this context it is important that the backed-up data originate from an PLC cycle.	The entire program management, i.e. which data are to be saved under what preconditions, must be handled by the programmer.
The PLC cycle time is not affected.	Because the process of writing to static RAM is considerably slower than that of writing to dynamic RAM, the PLC cycle is extended. Accordingly, the residual areas must be defined as small as possible.
Conclusion: With a PC with UPS data backup can be assured in case of a power failure.	Conclusion:Absolute data security cannot be ensured despite a cyclical and request-specific data backup procedure. Independent of the position in the program where processing is currently taking place, a power failure will cause an instant system stop without the backup of residual data.

3.3.4 Residual operation

The decision of residual/non-residual operation takes place in the OM2 /DW2, Bit2: “Residual if possible”.

In **residual operation**, the statuses of the designated residual operands are retained after a STOP/RUN and shutdown operating mode change.

Without special arrangements in the OM2 or the system area this means for the backup of residual data in the static RAM (osa master P-L/XL) or in the directory on the local hard disk (PNC):

- The upper half of the marker range M4096 through M8191 is residual.
- The upper half of the counters Z128 through Z255 is residual.
- The upper half of the timers T128 through T255 is residual.
- The upper half of the data fields DF16384 through DF32767 is always residual.
- Data modules marked with residual ID are always residual.

The user can shift the so-called ”residual limits” as desired. To this end, both OM2 and system area provide appropriate measures.

3.3.5 Non-residual operation

The decision of residual/non-residual operation takes place in the OM2 /DW2, Bit2: “Residual start if possible”.

In **non-residual operation**, a STOP/RUN operating mode change or Power-Off/On cycle will be followed by clearing all of the following:

- All markers
- All timers
- All counters

This occurs even before processing any startup OMs.

The entire data field, DF0 through DF32767, is always residual, regardless of the position of the residual switch.

☞ **For the backup of a data field in static RAM (osa master P-L/XL) or in the mounted directory of a hard disk (PNC) the defined residual limit (default OM2 or system area) applies.**

☞ **For the backup of data modules into static RAM (osa master P-L/XL) or into the mounted directory of a hard disk (PNC) only those DBs are taken into account that are marked with the residual ID (E) in the symbol file.**

3.3.6 Buffer failure, data backup fault

Buffer failure

The special marker SM20.1 buffer failure indicates a backup fault in the saving of residual data into static RAM of the osa master P-L/XL hardware or into the directory of the local hard disk (PNC).

The marker is set in the following cases:

- If during the run up of the iPCL it is recognised that no correct backup of residual data into static RAM of the osa master P-L/XL hardware or into the directory of a hard disk (PNC) was possible either during program processing or at shutdown of the last control cycle. The special marker is set while processing a startup OM and is reset once the PLC startup has concluded.
- During program loading or post-loading, the maximum permitted number of DBs with residual ID was exceeded. The special marker remains set during program processing, and is reset while loading, provided that the maximum permitted number of DBs with residual ID is maintained.
- The cyclical backup of residual data in the I/O state was not correctly executed, the maximum permitted number of DBs with residual ID was exceeded subsequent to online modifications. The special marker remains set during program processing.

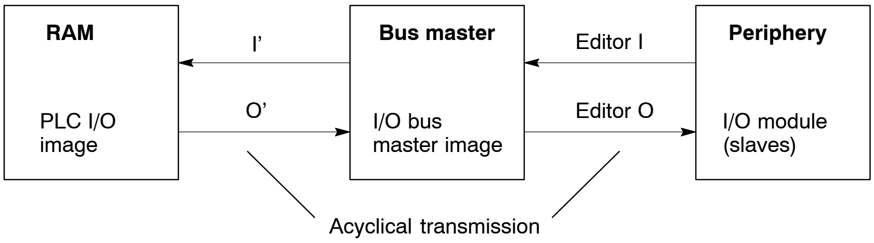
☞ **If the special marker SM20.1 is set then the backup of all residual operands is declined both cyclically and via PLC command into the static RAM of the osa master P-L/XL hardware or into the directory of the local hard disk (PNC) but the iPCL continues to RUN. The interpretation of the SM20.1 and/or of system area word S116 permits error handling.**

Data backup error

The special marker SM20.5 (data backup error) is set when, at the time of shutdown, the backup of residual data to the hard disk was faulty. The special marker is set during the processing of the startup OM5 power-on and is reset prior to the PLC startup OM1.

4 Peripheral Operation

The connection to the periphery is always via the PROFIBUS-DP. The PLC I/O data are transferred in I/O state or via a command to the image of the field bus master. The configured I/O modules (slaves) are serviced from there.



PROFIBUS-DP: Editor I/O

The bus master creates diagnostic tables on the basis of the I/O configuration list. The error messages and error diagnostic functions generated in this manner depend on the bus system being used, and must be evaluated with the aid of the bus-specific software tools.

4.1 Data exchange machine <—> PLC

The data exchange both to the machine (PROFIBUS-DP) and the NC (bit interface) is as follows:

In the PNC:

- The machine and NC inputs (bit interface) are read at the start of the PLC process.
- The machine and NC outputs (bit interface) are given out at the end of the PLC process.

In the osa master P-L/XL:

- As determined by the DCIO, for the osa master P-L/XL both the machine inputs and the machine outputs (PROFIBUS-DP) are replaced at the start of the PLC process.

☞ **The consistency requirements of the slaves are maintained.**

4.2 PROFIBUS-DP

Configuration

The I/O configuration for the PROFIBUS-DP is accomplished with the aid of the **WinDP** Configuration & Diagnostic Tool.

Data exchange

The data exchange between bus master image and peripheral devices is limited to those slaves that have been configured.

Data consistency

Data consistency is maintained only for those bus stations that have been appropriately configured. The data width depends on the default values taken from the device specification files.

Peripheral errors

The PROFIBUS-DP field bus features a comprehensive diagnostic system whose messages are made available to the PLC by the bus master. The WinDP software also incorporates the corresponding diagnostic system.When peripheral errors have been remedied, the PROFIBUS-DP restarts automatically.

Properties

General:

- Max. 124 slaves
- Max. 244 bytes each inputs and outputs per slave (max. 122 bytes consistent inputs or outputs)
- PROFIBUS-DP baud rates can be set to between 9.6 Kbits/s. and 12 Mbits/s.

Only PNC:

- PROFIBUS-DP V1
- Max. of 8Kbytes each inputs and outputs

Only osa master P-L/XL:

- PROFIBUS-DP protocol after EN50170
- Max. of 256 bytes each inputs and outputs

5 Programming Basics

Programmable memory controllers (iPCL) process a program whose code describes the controller task. This is accomplished with the use of special programming languages that can be represented and printed out in various modes.

5.1 Programming

Instruction List (IL):

The IL comprises a text-based programming language in which the controller tasks are written in assembler notation.

Structure of controller instructions:

Controller instruction					Line comment
Operations part	Operand attribute	Source operand	Destinatio n operand		
OPP	OPA	S-OPD	, D-OPD	;	Command description

Examples

```
U               I0.0
U       W       -Name   , O
L       B       O0      , B
T       D       C       , M12
MUL     W       1234    , D
```

Ladder Diagram (LD)

When using the LD representation method, the controller tasks are described by means of standard circuit diagram symbols.

Function Block Diagram (FBD)

With the programming language FBD logical connections are described using graphic symbols.

Sequential Function Chart (SFC)

The SFC represents a graphical programming interface, which is used to describe the sequentially processed machine tasks in the form of a cascade sequence. Before it can be loaded into the PLC, this representation is then translated into the executable IL programming language.

Structured Text (ST):

Structured text programming uses a text-based programming language in accordance with IEC 611313. Structured text is a high-level language which is easy to learn, and which facilitates compact formulation of programming tasks. Examples of its strong suits are the implementation of complex testing or regulating tasks.

5.2 Program Structure

In order to make PLC programs comprehensible and easy to read, structured programming is used in the PLC. Programs are divided into functionally associated program sections. To achieve structural clarity, various types of program modules are available, each handling specific tasks. Program processing can be cyclical, time or event driven.
An exemplary program structure is shown in section 5.6.4.

5.3 Module Types

The controller utilizes the following module types:

- Organization modules (OM)
- Program modules
- Data modules
- APS modules

All modules are activated by being invoked by the PLC program. This can occur unconditionally or contingent upon a condition. A condition may be the result of a logical or compare function or an arithmetic operation.

5.3.1 Organization modules (OM)

The organization modules perform all administrative or management functions for the controller program. Although they are programmed in the same manner as the program modules, only the system program invokes organization modules. All organization modules make use of the full instruction set of the PLC. There is no limitation to module size.

Each organization module is processed only subsequent to a defined condition; it cannot be called in the course of program processing.

Organization modules can be divided into 7 functional groups:

OM1	Program module that is called cyclically by the system program, and that can be used as a distribution module for the overall program.
OM2	Non-executable definition module (initialisation table) containing definitions for the controller system (residual limits, etc.) that are declared by modifying certain table entries.
OM5, OM7	Start-up modules that process various program sequences during a controller power-up or restart.
OM8	Module that is called upon shutdown; here the application can be brought to a defined state.
OM9	Error module that processes reactions when program errors occur.
OM18-25	Time-controlled processing (time scale can be defined in OM2).
OM30-63	Reserved.

The OM1 module must be concluded with either the EP (end of program) or EM (end of module) instruction to ensure subsequent processing of the input/output cycle (I/O state). With the exception of the OM2, all other organization modules can be concluded with either EP or EM, depending on the respective tasks being carried out.

5.3.2 Program modules

The program modules (PM) contain program segments that are technically and functionally interrelated. From within program modules, any number of additional program modules and data modules may be called. In addition, all program modules have access to the entire command set of the PLC. The modules are not subject to a size limit.

As a rule, program modules are concluded with an End of Module (EM) instruction. If end of program (EP) is used, after it has been processed, a program end follows and the I/O cycle is carried out. Then further program processing begins again with the OM1.

Due to the option of parameterisation, the program modules may be written independently of absolute operands. During the module call-up, the operands required for the current processing task are transferred to the program module in the form of parameter values.

The following parameters can be declared:

- Input parameters: Operands, constants and modules
- Output parameters: Operands
- I/O parameters: Operands

5.3.3 Data modules

The data modules (DM) serve as storage areas for all fixed and variable values and text blocks that are used by the program. Therefore, during PLC program processing, the user has the option of always keeping two data modules enabled, each of which provides up to 512 bytes of memory capacity.

The following applies to the processing of data modules:

- Before their respective data may be accessed, the data modules must be enabled from within the program by means of module call instructions (i.e., CM for the 1st DM, and CX for the 2nd DM).
- Within a given organization module (OM) or program module (PM), the data modules remain current until other data modules are enabled by the program.
- After the return to the primary module, the data modules active at the time of the call-up of the base module are again activated.
- When the OM1 (cyclical program processing), and the start-up modules OM5 and OM7 are called, no data module is active as yet.

5.3.4 APS modules

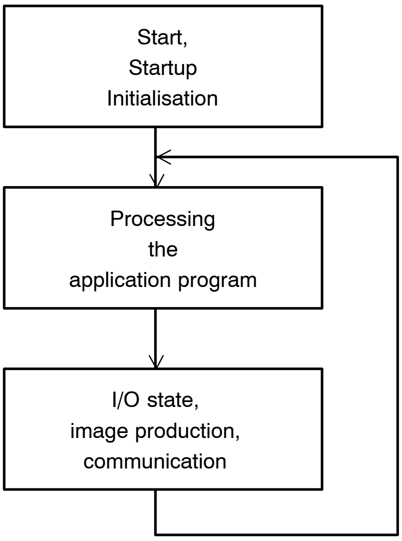
The functions of the APS modules (PLC application modules) are integrated into the firmware and the modules themselves merely contain the frame for the actual function calls. This ensures that the APS functions always fit the NC software. The mandatory call of the B01APSMN module at the start of the OM1 is also dropped as this function is carried out within the firmware at the start of PLC processing.

☞ **In the PNC no access to the serial interfaces via the APS modules are supported.**

☞ **If an existing program is taken over by an ICL project, the ICL project APS modules have to be replaced by iPCL APS modules.**
The integration of the program modules NcsLibW.pxh, IclLib.pxl, B01Apsmn.pxl and B06Lgana.pxl can be ignored.

5.4 Program Processing

The application program is processed cyclically and can be interrupted by time- or interrupt-controlled instructions.



Cyclical program processing

Once the iPCL has been initialised successfully, the actual program cycle begins in OM1 with the first command of the application program. The cycle time is measured from and until this point in time.

Subsequent to program processing, the processing of inputs and outputs and servicing of communication partners occurs before the cyclical processing continues.

Time-controlled program processing

In the course of cyclical program processing, the program sequence can be interrupted by elapsed times that can be defined in the time matrix. In this process, interruption points are only module changes (calling a data module does not rate as a module change). Program processing branches into an OM that is directly assigned to time-controlled processing, processes the program contained therein, and then returns to the interruption point.

In the event that the user avails himself of program module calls from within time OMs, he should disable any other time-controlled processes. See also Section Fehler! Es wurde kein Textmarkenname vergeben.5.16.

5.5 Time Monitoring

The entire program processing, i.e. the PLC cycle, is subject to time monitoring. Cycle time monitoring is used for this.

Cycle time monitoring comprises a security function that can be individually adjusted. Appropriate selection functions are provided in the OM2 initialisation table (see Section 5.7). If the OM2 is not linked to the controller program, this time will have a default value of 1.5 sec.

5.6 I/O state

The I/O state is always started after an EP (end of program) instruction, and processes the image update for peripheral operation, the processing of fixations and that of times / timers.

5.6.1 Fixing inputs, outputs & markers

The fixation imposes a fixed status mask on inputs, outputs and markers. The resulting fixation masks are placed over the I/O images and markers in each I/O state.

The fixation data are saved in a file in the controller filing system and are reloaded from there on startup. This file is updated with all changes to the fixation masks.

The resetting of fixations can be carried out via the program unit (WinSPS) or via PNC control.

The special marker SM20.4 indicates whether a fixation is active, i.e., at least one bit is fixed.

Fixed inputs

Prior to entering the OM1 of the PLC program, the loaded status (input image) is covered by the fixation mask. As a consequence, all input queries return the status taken from the fixation mask as long as they have not been changed by the PLC program.

Fixed outputs

Prior to the data exchange with the machine, the output status (output image) is covered by the fixation mask. As a consequence, all process outputs have the status imposed by the fixation mask.

Fixed markers

Prior to entering the OM1 of the PLC program, the status of the markers from the preceding PLC cycle is covered by the fixation mask. However, the queries within the PLC program will return the fixed status only until the program overwrites them.

5.6.2 Updating timers

Depending on the selected time matrix (resolution), the timers are updated also during the I/O state. This means that the accuracy of the timer loops with respect to the selected time matrix amounts to plus one PLC cycle (max.) including the I/O state.

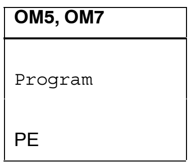
5.6.3 Cyclical processing

See Section 5.4

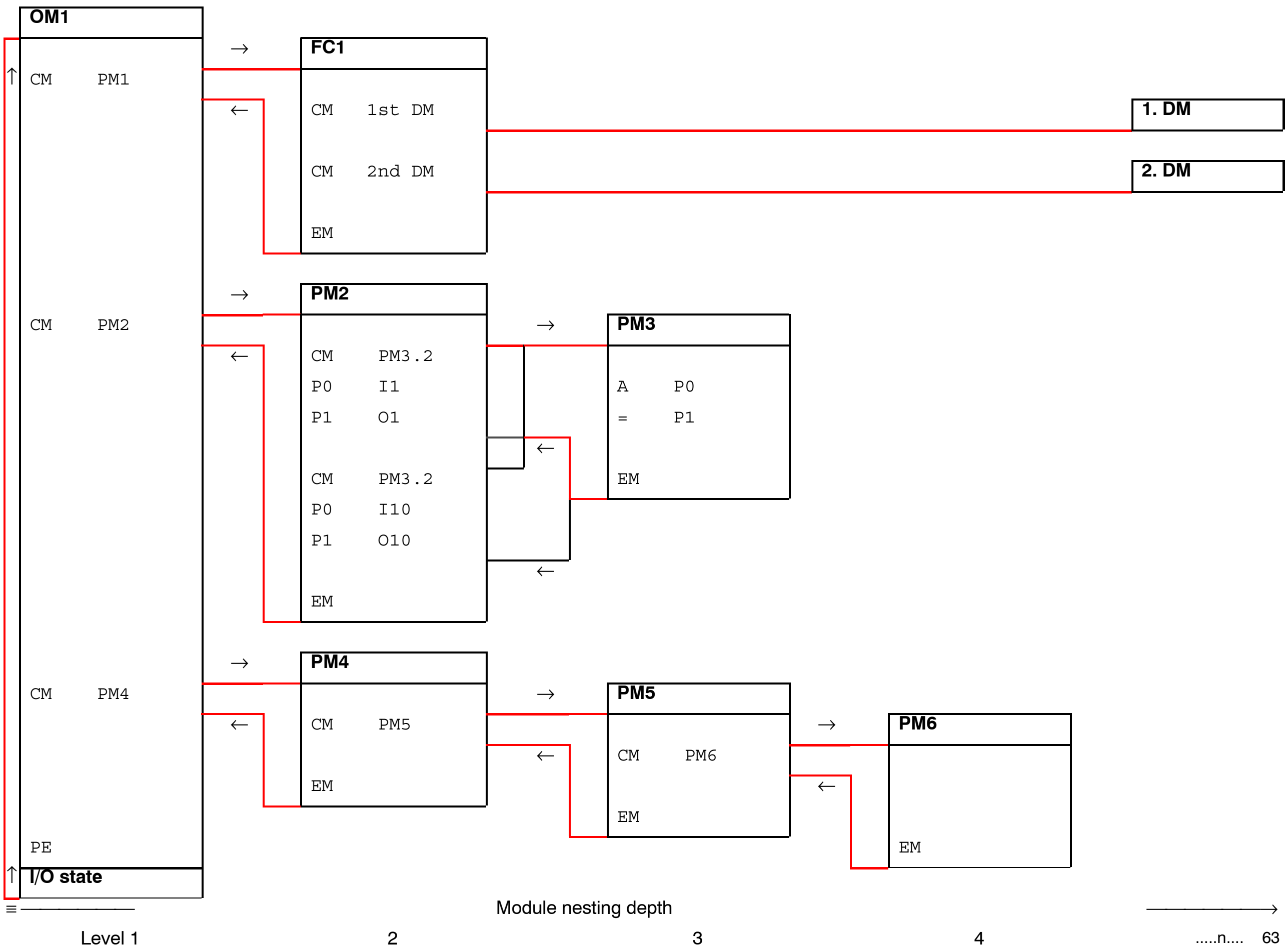
5.6.4 Application program structure

With the aim of providing a clear overview of the basic organization of program management, the following diagram shows an example of the program structure.

Program startup, one-time only



Program processing, cyclical



Time-controlled program processing

Processing always commences subsequent to the change of module (not module call) that follows the expiry of the associated time interval.

OM18-OM25
Program
EM

Program processing subsequent to a program error

Processing always occurs upon the occurrence of the triggering criterion.

OM9
Program
PE

5.7 Initialisation table OM2

The OM2 is a system initialisation table that can be linked to the PLC program as required. You will find a pre-configured module named OM2iPCL on the WinSPS path of your programming unit.

An PLC program working without an OM2 utilizes pre-selected default values that are sufficiently useful for many applications.

Deviations from the pre-selected system defaults are declared in the OM2 through manipulation of the entered values. It is essential that the user neither removes nor adds DEFW instructions.

This may be used for example, to shift residual limits, set cycle time limits, etc.

The time matrix definition for the time OMs is also handled in the OM2.

The declarations and definitions stored in the OM2 are adopted by the system upon Power-ON or in the case of a STOP/RUN command, even before processing a startup OM that may be present; a part of the OM2 is copied into the system area.

The following printouts of an OM2 exemplify all options of exercising control over the system initialisation:

5.7.1 Printout of the OM2iPCL

```
;*************************************************************************
;***                                                                   ***
;***         I N I T I A L I Z A T I O N    T A B L E                  ***
;***                                                                   ***
;***                       'i P C L'                                   ***
;***                                                                   ***
;*************************************************************************;
*** Last modification: 21.03.01, be                                     ***
;*************************************************************************
;
;*************************************************************************
; OM2 : iPCL  Initialisation table
;*************************************************************************
;
;      - Must be integrated into each application program;
;        that uses different default settings.
;
;     - If no OM2 entry is made in the symbol file,
;        default settings will be used.
;
;        I M P O R T A N T   N O T E , please observe in any case
;        ============================================================
;
;        EACH change of data words (W) in forbidden address ranges
;        ====
;        may result in undefined PLC system performance.
;
;*************************************************************************
;

;DW 1:  free
;--------------
DEFW  W  16#0000
;
;DW 2:  Initialisation flag (entries permitted)
;---------------------------------------------------------------------------
;     Entry 0 = DO NOT test or execute function
;     Entry 1 = Verify and/or execute function
;
DEFW  W  2#0000000000000000
;          *******|***||||*       *: not used
;                 |   |||+------- Check nominal cycle time
;                 |   ||+-------- Residual start if possible
;                 |   |+--------- Suppress cycle time monitoring
;                 |   |           during startup
;                 |   +---------- Max. I/O range for MMIMADAP diagnostics
;                 +-------------- Copy data module to data buffer
;

;DW 3:  System settings (entries permitted)
;---------------------------------------------------------------------------
;     Entry 0 = DO NOT test or execute function
;     Entry 1 = Verify and/or execute function
;
DEFW  W  2#0000000000000000
;          *********|||||||        *: not used
;                   ||||||+------ Markers        \  residual areas for
;                   |||||+------- Timers          \ for cyclical backup
;                   ||||+-------- Counters         \static RAM, as per
;                   |||+--------- Data field       /defined residual
;                   ||+---------- Data buffer     / limits
;                   |+----------- Data modules   /
;                   +------------ Cyclical backup of marked areas/ranges.
```

```
;
;DW 4:  free
;--------------
DEFW W  16#0000
;

;DW 5:  Maximum cycle time (entries permitted)
;----------------------------------------------------------------------------
;     Entries from 1 to 150 in multiples of time base 10 ms possible
;     (10 ms to 1500 ms) for cycle time monitoring.
;     Function execution at DW2 / Bit 1 = 1.
;
DEFW W  150
;

;DW 6:  Copy data module to data buffer (entries permitted)
;----------------------------------------------------------------------------
;     Entry of 0 - 1023 (data module number 0 - 1023) possible.
;     (Function execution at DW 2 / Bit 8 = 1).
;
DEFW W  0
;

;DW 7:  Number of first residual time (entries permitted)
;----------------------------------------------------------------------------
;     Entries from 0 to 256 are possible
;     128 = Residual  for timer loops T128 through T255
;     256 = No residual
;
DEFW W  128
;

;DW 8:  Number of first residual counter (entries permitted)
;----------------------------------------------------------------------------
;     Entries from 0 to 256 are possible
;     128 = Residual for counters Z128 to Z255
;     256 = No residual
;
DEFW W  128
;

;DW 9:  Number of first residual marker (entries permitted)
;----------------------------------------------------------------------------
;     Entries from 0 to 8192 are possible
;      4096 = Residual from marker byte M4096/marker bit M4096.0;
;             definition of residual limit via byte addresses
;      8192 = No residual
;
DEFW W  4096
;
;$P

;DW 10: First residual address in data buffer (entries permitted)
;----------------------------------------------------------------------------
;     Entries from 0 to 512 are possible
;      256 = Residual from data buffer byte DP256
;      512 = No residual
;
DEFW W  256
;
;
 ;    Definition of timer OMs (entries permitted)
;     ========================
;     Entries as multiplier of base time 10 ms from 1 to 65535
;        possible
;     e.g. 0  = no timer-based processing
;          11 = 11 x 10 ms = 110 ms interval of processing time
```

```
;
;DW 11: time OM18
;----------------
DEFW  W  0

;DW 12: time OM19
;----------------
DEFW  W  0

;DW 13: time OM20
;----------------
DEFW  W  0

;DW 14: timer OM21
;----------------
DEFW  W  0

;DW 15: time OM22
;----------------
DEFW  W  0

;DW 16: time OM23
;----------------
DEFW  W  0

;DW 17: time OM24
;----------------
DEFW  W  0

;DW 18: time OM25
;----------------
DEFW  W  0
;
;$P

;DW 19 - DW 32: empty
;--------------------
DEFW  W  16#0000    ;DW19
DEFW  W  16#0000    ;DW20
DEFW  W  16#0000    ;DW21
DEFW  W  16#0000    ;DW22
DEFW  W  16#0000    ;DW23
DEFW  W  16#0000    ;DW24
DEFW  W  16#0000    ;DW25
DEFW  W  16#0000    ;DW26
DEFW  W  16#0000    ;DW27
DEFW  W  16#0000    ;DW28
DEFW  W  16#0000    ;DW29
DEFW  W  16#0000    ;DW30
DEFW  W  16#0000    ;DW31
DEFW  W  16#0000    ;DW32

;DW 33: First residual address in data field for backup to
;        static RAM (entries permitted)
;---------------------------------------------------------------
;     Entries of 0 and 32768 possible
;      16384 = Residual from data field byte DF16384 in static RAM
;     32768 = No residual in static RAMLimit applies only to backup
;        into static RAM; this areatakes precedence over the data field,
;        the entire remainderof which is backed up to hard disk for
;        residual storage;
;
DEFW  W  0
;
;DW 34 - DW 101: empty
;------------------
DEFW  W  16#0000    ;DW34
```

```
DEFW  W  16#0000    ;DW35
DEFW  W  16#0000    ;DW36
DEFW  W  16#0000    ;DW37
DEFW  W  16#0000    ;DW38
DEFW  W  16#0000    ;DW39
DEFW  W  16#0000    ;DW40
DEFW  W  16#0000    ;DW41
DEFW  W  16#0000    ;DW42
DEFW  W  16#0000    ;DW43
DEFW  W  16#0000    ;DW44
DEFW  W  16#0000    ;DW45
DEFW  W  16#0000    ;DW46
DEFW  W  16#0000    ;DW47
DEFW  W  16#0000    ;DW48
DEFW  W  16#0000    ;DW49
DEFW  W  16#0000    ;DW50
DEFW  W  16#0000    ;DW51
DEFW  W  16#0000    ;DW52
DEFW  W  16#0000    ;DW53
DEFW  W  16#0000    ;DW54
DEFW  W  16#0000    ;DW55
DEFW  W  16#0000    ;DW56
DEFW  W  16#0000    ;DW57
DEFW  W  16#0000    ;DW58
DEFW  W  16#0000    ;DW59
DEFW  W  16#0000    ;DW60
DEFW  W  16#0000    ;DW61
DEFW  W  16#0000    ;DW62
DEFW  W  16#0000    ;DW63
DEFW  W  16#0000    ;DW64
DEFW  W  16#0000    ;DW65
DEFW  W  16#0000    ;DW66
DEFW  W  16#0000    ;DW67
DEFW  W  16#0000    ;DW68
DEFW  W  16#0000    ;DW69
DEFW  W  16#0000    ;DW70
DEFW  W  16#0000    ;DW71
DEFW  W  16#0000    ;DW72
DEFW  W  16#0000    ;DW73
DEFW  W  16#0000    ;DW74
DEFW  W  16#0000    ;DW75
DEFW  W  16#0000    ;DW76
DEFW  W  16#0000    ;DW77
DEFW  W  16#0000    ;DW78
DEFW  W  16#0000    ;DW79
DEFW  W  16#0000    ;DW80
DEFW  W  16#0000    ;DW81
DEFW  W  16#0000    ;DW82
DEFW  W  16#0000    ;DW83
DEFW  W  16#0000    ;DW84
DEFW  W  16#0000    ;DW85
DEFW  W  16#0000    ;DW86
DEFW  W  16#0000    ;DW87
DEFW  W  16#0000    ;DW88
DEFW  W  16#0000    ;DW89
DEFW  W  16#0000    ;DW90
DEFW  W  16#0000    ;DW91
DEFW  W  16#0000    ;DW92
DEFW  W  16#0000    ;DW93
DEFW  W  16#0000    ;DW94
DEFW  W  16#0000    ;DW95
DEFW  W  16#0000    ;DW96

;DW 97: MTB1 and MTB2 allocation (entries permitted)
;--------------------------------------------------
;     Entry 0 = DO NOT execute function
;     Entry 1 = Execute function
```

```
;
;     If the CAN processing is blocked then the
;     MTB1/MTB2 processing will not be carried out!
;
DEFW  W  2#0000000000100101
;         *********||*||*|       *: not used
;                  || || +------ CAN processing
;                  || |+-------- MTB1 processing
;                  || +--------- MTB2 processing
;                  |+----------- CAN actual assignment
;                  +------------ Suppress CAN error/warning
;
;DW 98: reserved
;-----------------
DEFW  W  16#0000
;
;
;DW 99: Start address of MTB1 information in the marker area
;-------------------------------------------------------------
;(entries permitted)
;     The MTB1 data occupy 20 successive bytes in the
;     marker area:
;                  Meaning:   16 bytes input data
;                             4 bytes output data
;     The function must be enabled in DW 97.
;     Entries from K0D to K6124D (M0 to M6124) are possible.
;
DEFW  W  6100


;DW 100: Start address of MTB2 information in the marker area
;--------------------------------------------------------------
;(entries permitted)
;     The MTB2 data occupy 20 successive bytes in the
;     marker area:
;                  Meaning:   16 bytes input data
;                             4 bytes output data
;     The function must be enabled in DW 97.
;     Entries from K0D to K6124D (M0 to M6124) are possible.
;
DEFW  W  16#0000
;
;DW 101: Start address of the CAN actual assignment in the marker area
;------------------------------------------------------------
;     ;(entries permitted)
;     The CAN actual assignment takes up 2 bytes in the marker area
;     and contains the following information:
;
;         0000000000000000B
;         ************||**       *: not used
;                     ||
;                     |+-------- MTB 1  recognized
;                     +--------- MTB 2  recognized
;
;The function must be enabled in DW 97.
;  Entries from K0D to K6124D (M0 to M6124) are possible.
DEFW  W  6122
;$P
;
;     !!!     Internal system memory data     !!!
;     ===========================================
;
;The following default settings must not be changed.
;  ==================================================================
;Default value for data words DW 102 - DW 127 = 16#0000
;______________________________________________________
DEFW  W  16#0000    ;DW102
```

```
DEFW  W  16#0000    ;DW103
DEFW  W  16#0000    ;DW104
DEFW  W  16#0000    ;DW105
DEFW  W  16#0000    ;DW106
DEFW  W  16#0000    ;DW107
DEFW  W  16#0000    ;DW108
DEFW  W  16#0000    ;DW109
DEFW  W  16#0000    ;DW110
DEFW  W  16#0000    ;DW111
DEFW  W  16#0000    ;DW112
DEFW  W  16#0000    ;DW113
DEFW  W  16#0000    ;DW114
DEFW  W  16#0000    ;DW115
DEFW  W  16#0000    ;DW116
DEFW  W  16#0000    ;DW117
DEFW  W  16#0000    ;DW118
DEFW  W  16#0000    ;DW119
DEFW  W  16#0000    ;DW120
DEFW  W  16#0000    ;DW121
DEFW  W  16#0000    ;DW122
DEFW  W  16#0000    ;DW123
DEFW  W  16#0000    ;DW124
DEFW  W  16#0000    ;DW125
DEFW  W  16#0000    ;DW126
DEFW  W  16#0000    ;DW127
DEFW  W  16#0000    ;DW128

;***************************************************************************
EM
```

5.8 Module reference list

The module reference list comprises a Table of Contents listing the modules integrated in the PLC program. The list contains information about module existence, module size and module start address.

To extract this data, special instructions are available for the user.

☞ **The instructions used to verify module existence, module size and module start address of OMs and PMs can be used only with the WinSPS v3.0 and higher.**

5.9 Module existence

Example:

```
; Check module existence
;-------------------------
; Checks whether the modules OM8, DM8, and FC8.
; exist

; direct addressing
U       OM8   ; OM8 exist?
U       DM8   ; DM8 exist?
U       PM8   ; PM8 exist?
; indirect addressing
L   D   8,A   ; load module no. in register A
U       OM[A]; OM8 exist?
U       DM[A]; DM8 exist?
U       PM[A]; PM8 exist?
```

5.10 Module size

Example:

```
; read module size
;---------------------
; Extracts module lengths of modules OM8, DM8, and PM8.

; direct addressing
L   D   OM8,A; size of OM8 in reg. A
L   D   DM8,A; size of DM8 in reg. A
L   D   PM8,A; size of PM8 in reg. A
; indirect addressing
L   D   8,A   ; load module no. in register A
L   D   OM[A],B    ; size of OM8 in reg. B
L   D   DM[A],B    ; size of DM8 in reg. B
L   D   PM[A],B    ; size of PM8 in reg. B
```

5.11 Module start address

Example:

```
; Read module start address
;---------------------------
; Extracts module start addresses for modules OM8, DM8, and
PM8.
; direct addressing
L   D   &OM8,A  ; start address of the OM8 in reg. A
L   D   &DM8,A  ; start address of the DM8 in reg. A
L   D   &PM8,A  ; start address of the PM8 in reg. A

; indirect addressing
L   D   8,A   ; load module no. in register A
L   D   &OM[A],B   ; start address of the OM8 in reg. B
L   D   &DM[A],B   ; start address of the DM8 in reg. B
L   D   &PM[A],B   ; start address of the PM8 in reg. B
```

5.12 Module header

The module header contains information about the following:

- Module start address
- Module size
- Module version number, generated by the WinSPS module header editor
- Length of module name (currently = max. 8)
- Module name in string notation.

The user can employ a special instruction to evaluate this data. The function of this instruction is explained in the following example.

☞ **The commands for checking module headers are available from the WinSPS version onwards.**

Example:

```
; Write module header contents on marker
;------------------------------------------
; 20 bytes of the FC100 module header shall be stored
;from marker M20 onwards.
;
; Number of bytes to be read must be in register C.L   D   20,C

; Writing 20 bytes of header information onto an operand.
; The start address of the operand must be a multiple
; of 4 due to the double-word processing.
FC    D    FC100,M20   ; store 20 byte header contents of the FC100
from M20
;                     4 bytes   (M20-M23): Start address
;                     4 bytes   (M24-M27): Size in bytes
;                     2 bytes   (M28+M29): Version no. from header
;                     1 byte    (M30):     Length n of module name
;                     8+1byte   (M31-M39): Module name string with
;                                         '\0' at the end.
;                     2 bytes            : PXL/PXO code:
;                                          1 = secret
;                                          0 = not secret
```

The user can utilize this command sequence to read the module header information of OMs, PMs and DMs. It should be noted that DMs do not feature version identifiers in the module header, i.e., the respective bytes have a content = 0.

5.13 OM9 error module

This module is invoked once only in the event that a program error is noted that would normally cause an immediate stop of the central processing control unit. To serve the intended purpose, it must be integrated into the PLC program.

The triggering criteria are defined errors that can be interpreted by setting a special marker bit in SM14 / SM15 and in SM28 / SM29.

Upon calling the OM9, the cycle time monitoring function is restarted with the defined value (definition in OM2 or default value of 1.5 sec). While the module is being processed, countermeasures for possible error occurrences can be programmed.

For example, certain data, including the special error markers, can be moved to non-volatile areas.

Once the OM9 error module has been processed, the PLC enters STOP mode.

5.14 Fixation

The PLC provides the option to fix operands.

In contrast to the ”Control” programming device function, this option can be used to fix operands permanently to specific bit statuses or values.

Operands suitable for fixation:
- Inputs
- Outputs
- Markers

Residuals of fixation

An existing fixation is retained in the following cases:
- Always after a STOP/RUN change in operating mode.
- After a new load.
- Always after a Power-Off/On cycle.

5.15 Parameterized Modules

When a program module is called up, up to 63 parameter values can be transferred. The number of parameters transferred is specified in the module call-up command. Then the parameters follow, starting with P0.

Example of parameter transfer

```
DEF       E0.0,-Start
DEF       M0,-Target value
DEF       M2,-Actual value
DEF       A0.0,-Target_actual
DEF       A0.0,-No result

;
BA        -TARGET_ACTUAL,5
;
;                               +---------------------+
P0        -Start            ; | BOOL    VAR_INPUT  | Signal for function start
P1    W   -Target value     ; | WORD    VAR_IN_OUT | expected number
P2    W   -Actual value     ; | WORD    VAR_INPUT  | actual number
P3        -Target_actual    ; | BOOL    VAR_IN_OUT | target value reached
P4        -No result        ; | BOOL    VAR_OUTPUT | no valid reading
;                               +---------------------+
```

Utilization of parameters in called–up module:

```
+--------------------------------------------------------------------------------+!Parameter
header
+--------------------------------------------------------------------------------+
   P0      BOOL    Start            VAR_INPUT       Signal for function start
   P1      WORD    Target value     VAR_IN_OUT      expected number
   P2      WORD    Actual value     VAR_INPUT       actual number
   P3      BOOL    Target_actual    VAR_IN_OUT      target value reached
   P4      BOOL    no result        VAR_OUTPUT      no valid reading

+ ------------------------------------------------------------------------------+
! Program module file
+ ------------------------------------------------------------------------------+
     ; Compare values
1    U        -Start                    P0      Signal for function start
2    SPI      no comparison
3    L     W  -Target value,A           P1      expected number
4    VGLA W   -Actual value,A           P2      actual number
5    U        Z                               ; Result=0 --> values are equal
6    =        -Target_actual            P3      target value reached
7    R        -No result                P4      no valid reading
     no comparison:
     ; Delete compare result
8    UN       -Start                    P0      Signal for function start
9    R        -Target_actual            P3      target value reached
10   S        -No result                P4      no valid reading

11   EM
```

5.16 Time-controlled program processing

iPCL provides the option of time-controlled program processing.

☞ **For time-controlled processing 8 timer OMs are provided that interrupt the program at predefined intervals to activate one of these modules. The timer resolution (matrix) is defined in the OM2.**

A timer OM is called up if:

1. The designated time interval has expired and
2. a change of module has been reached.

Defined module changes are an executed module call, as well as an end of module. Neither a DM call-up nor an EP instruction is considered a change of module.Within the group of timer interrupts, the highest priority is given to the interrupt that is assigned to the lowest OM number.

OM18 = highest priority, OM25 = lowest priority

☞ **Because some programs utilize the register contents across module boundaries (e.g., MADAP with the KETTEPCL program module), the register contents should always be backed up upon entry into a timer OM, and again updated prior to the end of module (PUSH/POP).**

Commands for handling timer interrupts

The time-controlled interrupts (TI) are assigned an interrupt mask. This mask can be read and written to with the use of the TIM and LIM instructions, respectively. Each possible timer interrupt corresponds to one bit in this mask. When a bit is set, this means that the respective interrupt has been enabled; when the bit is not set, the interrupt is disabled.

To perform the actual enabling of the interrupts declared in the mask, the additional instruction EAI (Enable All Interrupts) must be issued. A general disabling of the interrupts without influencing the mask is accomplished with the DAI (Disable All Interrupts) instruction.

Incoming interrupts cause an entry in the corresponding interrupt register even in cases where the respective interrupts have been masked. Here again, a bit is assigned to each timer interrupt.

If the interrupt is executable, i.e. enabled, calling the interrupt OM automatically deletes the bit in the interrupt register.

When the interrupt is disabled, the bit remains in the register, and the interrupt awaits its being enabled.

The interrupt register can be loaded using the LAI (Load All Interrupts) instruction, and active interrupts can be deleted with the RAI (Reset All Interrupts) instruction.

A change of operating mode, i.e. STOP/RUN or Power-Off/On, deletes all active interrupts.

By default, all time controlled interrupts are enabled.

During the startup procedure, i.e. processing of OM5 and OM7, all interrupts remain disabled.

5.17 Application stack

The application stack (AST) comprises a pushdown-pop-up memory stack with a storage depth of 256 double words, using FILO (first-in-last-out) processing.

The PUSH and POP instructions facilitate a word-by-word data transfer between the registers and the contents of the application stack.

Example:

```
PUSH    A  ;Shift contents of register A to applic. stack
PUSH    B  ;Shift contents of register B to applic. stack
PUSH    C  ;Shift contents of register C to applic. stack
PUSH    D  ;Shift contents of register D to applic. stack

POP    D  ;Load uppermost value from applic.stack into Reg.D
POP    C  ;Load uppermost value from applic.stack into Reg.C
POP    B  ;Load uppermost value from applic.stack into Reg.B
POP    A  ;Load uppermost value from applic.stack into Reg.A
```

In the event of an application stack underflow, special marker bit S28.4 will be set to ON.

In the case of an application stack overflow, special marker bit S28.5 will be set to ON. Both application stack (AST) underflow and overflow conditions will cause the central processing module to enter STOP mode, returning an error message indicating the cause of the error.

The application stack is flushed after each EP!

6 iPCL addressing

6.1 Operand & module identifiers, module list

Abbrev.	Indexed	Operand	Access / Data width	Image update
A, B, C, D		General computingre-gisters	Bit, byte, word, double word, REAL,LREAL	
I	I[R]	Input	Image/Bit, byte, word, double word, REAL,LREAL	I/O state
O	O[R]	Output	Image/Bit, byte, word, double word, REAL,LREAL	I/O state
M	M[R]	Markers	Bit, byte, word, double word, REAL, LongREAL	
SM	SM[R]	Special marker	Bit, byte, word, double word, REAL, LongREAL	
T	T[R]	Timer	Bit (status), word (value)	
Z	Z[R]	Counters	Bit (status), word (value)	
D DX	D[R] DX[R]	Data word, 1st current DM Data word, 2nd cur-rent DM	Bit, byte, word, double word, REAL,LREAL	
DM	DM[R]	Data buffer	Bit, byte, word, double word, REAL,LREAL	
DF	DF[R]	Data field	Bit, byte, word, double word, REAL,LREAL	
S	S[R]	System data range	Bit, byte, word, REAL, LREAL	
P	P[R]	Parameter	Bit, byte, word, double word	
FI		FIFO	Max. 512 bytes	
TI		Time-controlled inter-rupt		
b#www		Constants	Bit, byte, word, double word, REAL,LREAL	
DM	DM[R]	Data module	CM DMnn ; calls 1st DM	
PM	PM[R]	Program module	BX DMnn ; calls 2nd DM	

In the above enumeration, "R" is replaced by the register IDs "A", "B", "C" or "D".

Module list

iPCL manages the following modules:

Name	Function	Comment
OM1	Cyclical program processing	
OM2	Initialisation table	Refer to Section 5.7 "Initialisation Table"
OM5	Startup module after Power-ON	
OM7	Startup module after STOP/RUN	
OM8	Shutdown module	
OM18–OM25	Time-controlled modules	Matrix agreement in the OM2 or S18 – S32, lowest module no. = highest priority
OM42 – OM63	reserved	
PM0 – PM1023	Program modules	
DM0 – DM1023	Data modules	

6.2 Assignments in the special marker area

The iPCL features a special marker area with a size of 16-words i.e. SM0 through SM30. It contains essential information regarding system flags and PLC cycle time.

The unused addresses are reserved for internal system functions, and must not be changed.

Address	Contents		Comment
SM14	PLC program and system error messages:		
	Hex		
	12	Cycle time error	
	16	Module stack overflow	
	17	Application stack overflow	
	18	Application stack underflow	
	19	DM too short	
	1A	Operation code error	
	1B	Parameter error	
	1C	Parameter not found	
	1D	Address error, access to invalid address, e.g. transfer to constant or timer or actual counter value.	
	1E	Not available PB called up	
	1F	Not available DM called up	
	20	Halt command	
	21	Controller in STOP	
	22	Hardware error	
	23	”C” application error	
	24	”C” application warning	
	25	Re-entrant module call	
	26	Assignment list error	
	27	No PLC program	
	28	Error in call for peripheral driver	
	29	Error in installation of peripheral driver	
	2B	Not available Interr. OM	
	2C	Instruction not yet integrated	
	2D	Error in indirect jump	
	2E	Wrong operand number	
	2F	DM not active	
	30	Illegal DM size	
	31	Non-reproducible error	
	41	System software error	
SM16			
SM18			

Address	Contents		Comment
SM20	Bit		Read-only for entire bit field
	20.0	Trigger pulse upon each startup	
	20.1	Buffer failure	= 1, SNCI4: Memory fields to static RAM
	20.2	Flashing marker	
	20.3	Outputs disabled	
	20.4	Fixation active	
	20.5	Data backup error	= 1, PNC: Memory fields of the residual data on the hard disk
	20.6	Cold start flag	
	20.7	Trigger pulse after power ON or load	
SM22		Actual cycle time of last complete cycle	Read-only
SM24		Maximum measured cycle time	
SM26		Minimum measured cycle time	
SM28	Error word 1		All errors are read-only
	28.0	Addressing error	
	28.1	Parameter error	
	28.2	Non-existent module called	
	28.3	Module stack error	
	28.4	AST underflow	
	28.5	AST overflow	
	28.6		
	28.7		
SM29	Error word 1		
	29.0		
	29.1	Opcode error	
	29.2		
	29.3		
	29.4		
	29.5	No DM active	
	29.6	Group error message	For detailed information, refer to SM14.
	29.7	Cycle time error	
SM30	Auxiliary marker word		All errors are read-only
	30.0		
	30.1		
	30.2	Always 0	
	30.3		
	30.4		
	30.5		
	30.6		
	30.7		
SM31	Auxiliary marker word		
	31.0	Logical greater flag	Influenced only by CPL instruction
	31.1	Always 1	
	31.2		
	31.3		
	31.4		
	31.5		
	31.6	Carry flag, logical less when 1	Influenced only by CPL instruction
	31.7	Zero flag, logical equal when 1	Influenced only by CPL instruction

6.3 System area assignment

iPCL features a system area with a size of 512 bytes i.e. S0 through S511. It contains the system configuration data for the respective controller. Essential declarations made in OM2 are copied into the system area, and can thus be read by the PLC program.

To the extent deemed useful, the system declarations may be changed at runtime. This also includes the time intervals of time-controlled organization modules.

Segments of the system area are used by default function modules which make data available that is also used by other PLC program parts.

Example: Date and time.

The unassigned addresses in the system area are reserved for internal purposes, and must not be modified.

Address	Contents		Comment
S0	Initialisation flags like OM2_DW2		Writing in OM5 / OM7
S2	System settings like OM2_DW3		
S4	Error reaction like OM2_DW4		Writing in OM5 / OM7
S6	Maximum cycle time like OM2_DW5		Writing in OM5 / OM7
S8	DM to be copied like OM2_DW6		Read-only
S10	First residual time like OM2_DW7		Writing in OM5 / OM7
S12	First residual counter like OM2_DW8		Writing in OM5 / OM7
S14	First residual marker address like OM2_DW9		Writing in OM5 / OM7
S16	First residual data buffer address like OM2_DW10		Writing in OM5 / OM7
S18	Time interval OM18 like OM2_DW11		Transfer during startup and EP, possibly active timer must expire before new matrix is activated.
S20	Time interval OM19 like OM2_DW12		
S22	Time interval OM20 like OM2_DW13		
S24	Time interval OM21 like OM2_DW14		
S26	Time interval OM22 like OM2_DW15		
S28	Time interval OM23 like OM2_DW16		
S30	Time interval OM24 like OM2_DW17		
S32	Time interval OM25 like OM2_DW18		
S62	First residual data field address like OM2_DW33		
S64	Current processing time, in microseconds		Program run time OM1 start through I/O state end.
S66	Current processing time, in milliseconds		
S68	Max. processing time, in microseconds		
S70	Max. processing time, in milliseconds		
S72	Min. processing time, in microseconds		
S74	Min. processing time, in milliseconds		
S76	Min. processing time, in microseconds		RUN = READY contact closed
S100	Real-time:	Minutes / seconds	Read-only
S102		Day/ hours	Entry from operating system 0=So, 1= Mo,... , 6=Sa
S104		Year / month	
S106		----- Week day :	

Address	Contents	Comment
S114	Periphery status	See Section Periphery status
S116	Residual status	See Section iPCL startup characteristics
S124	I size	I/O information
S126	O size	SNCI4: 0.5 KbytesPNC 8 Kbytes
S128	Hardware / software version	
S151	Field bus type	DCIO PROFIBUS-DP (with SNCI4)
S152	Field bus type	PNC PROFIBUS-DP (with PNC)
 S240 .. S255	PROFIBUS-DP slave diagnostics BTN 15 .... 0 .... BTN 127 112	Bit state: 0 = Slave working error free 1 = Slave reports diagnostics (cannot be contacted or reports an error)
S510		

6.4 Periphery status

The periphery status word S114 provides an overview of the status of the bus master; it has the following format:

Bit		Description
0	BMF	Bus master error
1	KSD	Classified slave diagnostics: The KSD bit in the DP status word is the OR link of bits 8 to 13. The individual error types of the KSD are shown in bits 8 to 13 of the DP status word.
2	SD	System diagnostics: The DP standard differentiates between system diagnostics and slave diagnostics. System diagnostics comprise a bit field that indicates which slaves report diagnostics. In addition, there is a detailed diagnostic routine for individual slaves, the slave diagnostics.The SD bit in the DP status word represents the OR link of all system diagnostic bits. Therefore, when SD = 1, at least one slave reports diagnostics.
3		Reserved
4	Init	Init phase: Waiting until periphery is ready for operations, or until iPLC STOP time has elapsed.
5	BmClab	Bus master has switched DP bus to CLEAR status: BmClab = [SNE v SKF v SNB] & Error_Action_Flag = 1. The point in time for the restart after discontinuation of the BmClab causes can be controlled from the PLC program.
6	PgStop	Programming unit keeps DP bus in STOP state.
7	Active	Active ID: This bit must always be 1. If that is not the case, then there is a fatal error in the bus master software.
8	SNE	One or more slaves are not reachable on the bus.
9	SKF	One or more slaves report configuration errors.
10	DPS	One or more slaves report static diagnostics.
11	EXD	One or more slaves report extended diagnostics.
12	SNB	One or more slaves not ready for cyclical data exchange.
13	SF	One or more slaves report error of another type.
14		Reserved
15		Reserved

The bits **Init**, **BmClab**, **PgStop** are not relevant to the PLC program because, in the RUN state of the iPCL, they always have the value 0..

Bus master error (BMF)

This bit indicates that a bus master error has been detected.

KSD – Classified Slave Diagnostics

The KSD bit in the DP status word is the OR link of bits 8 to 13. The individual error types of the KSD are shown in bits 8 to 13 of the DP status word.

System diagnostics in accordance with DP standards (SD)

The DP standard differentiates between system diagnostics and slave diagnostics. System diagnostics comprise a bit field that indicates which slaves report diagnostics. In addition, there is a detailed diagnostic routine for individual slaves, the slave diagnostics.
The SD bit in the DP status word represents the OR link of all system diagnostic bits. Therefore, when SD = 1, at least one slave reports diagnostics.

Active ID

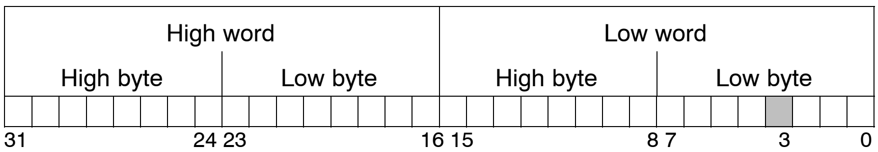
This bit must always be 1. If that is not the case then there is a fatal error in the bus master software.

6.5 Data formats

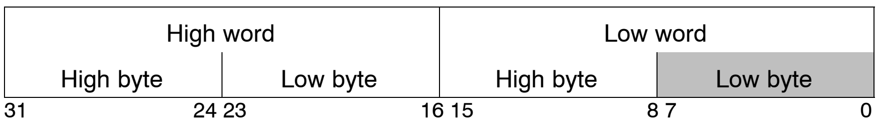
Bit, byte, word and double word can all be specified as **data formats** . In the **addressing** differentiation is made between:

- Load instruction
- Transfer instruction

Bit



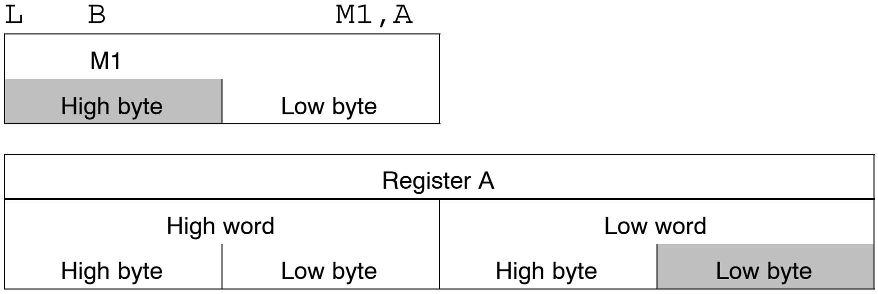
Byte = B



On loading, the source operand may be either the even-numbered (LOW) byte or the odd-numbered (HIGH) byte. In the case of the destination operand (register), the LOW byte is always addressed.

Example: Load command (byte): M1

```
L   B          M1,A
```



Example: Load command (byte): M2

```
L    B            M2,A
```

M2	
High byte	Low byte

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

On transfer the LOW byte is addressed in the source operand (register). The destination operand may be either the even-numbered (LOW) byte or the odd-numbered (HIGH) byte.

Example: Transfer command (word): M1

```
T    B            A,M1
```

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

M1	
High byte	Low byte

Example: Transfer command (word): M2

```
T    B            A,M2
```

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

M2	
High byte	Low byte

Word = W

High word		Low word	
High byte	Low byte	High byte	Low byte
31 – 24	23 – 16	15 – 8	7 – 0

An even-numbered or odd-numbered byte address may be specified for word processing during the load or transfer instructions.

Without exception, for the **load** instruction, the specified byte and the subsequent byte are loaded into the LOW word of the register (32-bit); the HIGH word of the register remains unchanged.

Without exception, for the **transfer** instruction, the specified byte and the subsequent byte are written from the LOW word of the register (32-bit).

Example: Load command (word): M2

```
L    W              M2,A
```

	M2
High byte	Low byte

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

Double word = D

High word		Low word	
High byte	Low byte	High byte	Low byte
31 … 24	23 … 16	15 … 8	7 … 0

Loading always requires the base byte and the following 3 bytes to be loaded into the specified register (32-bit).

Transferring always requires the specified register (32-bit) to be written to the base byte and the following 3 bytes.

Example: Load command (double word): M4

```
L    D              M4,A
```

			M4
High byte	Low byte	High byte	Low byte

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

Example: Transfer command (double word): M4

```
T   D          A,M4
```

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

			M4
High byte	Low byte	High byte	Low byte

Example: Transfer command (double word): M3

Error in the PG.

```
L   D           3,C
T   D         A,M[C]
```

Register A			
High word		Low word	
High byte	Low byte	High byte	Low byte

			M3
High byte	Low byte	High byte	Low byte

6.6 Register structure

The controller features 4 working registers, which can be addressed in a bit-wise, byte-wise, word-by-word or double word fashion. In this context, it should be noted that byte/word addressing always addresses the LOW-byte/word.

Working registers A, B, C, D

31	24 23	16 15	8 7 0
High word		Low word	
High byte	Low byte	High byte	Low byte

For operations that exceed the 32-bit format, the registers are combined to form permanent register pairs.

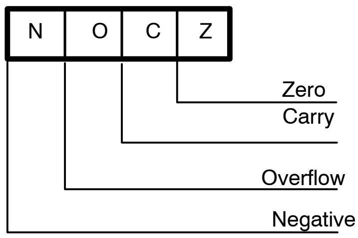
Working register pair A + B

31 24 23 16	15 8 7 0
Word 4 = HIGH word B	Word 3 = LOW word B
Word 2 = HIGH word A	Word 1 = LOW word A

Working register pair C + D

31 24 23 16	15 8 7 0
Word 4 = HIGH word D	Word 3 = LOW word D
Word 2 = HIGH word C	Word 1 = LOW word C

Status bits



6.7 Representation of constants

Data type		PLC service program WinSPS
Description	**Representation**	
UINT (unsigned integer)	Binary / dual, word	2#00000000_00000000 to 2#11111111_11111111
	Decimal, wordDouble word	0 to 655350 0 to 4294967295
	Hexadecimal, wordDouble word	16#0000 to 16#FFFF 16#00000000 to 16#FFFFFFFF
INT (signed integer)	Decimal, wordDouble word	–32768 to +32767 –2147483648 to +2147483647
Floating point REAL LREAL	Double word Quadword	$1.175494351e^{-38}$ to $3.402823466e^{+38}$ $2.2250738585072014e^{-308}$ to $1.7976931348623158e^{+308}$
Text, STRING(2)	ASCII, word double word	’AB’ ’ABCD’
Time value TVALUE	Time value (+time base r)r: 0 = 10 ms, 1 = 100 ms, 2 = 1 s, 3 = 10 s	T#10 ms to T#10230 s T#0.r to T#1023.r
TCP/IP addresses, ISTRING	Double word	”1.2.3.4”

6.8 Program module calls

	PLC service program WinSPS	
Program module / function call (IEC1131/3)	CM	PM

6.9 Jump instructions

	PLC service program WinSPS	
Jump instruction	JPx	label
Jump destination	label	

6.10 Bit- and module addresses

Operand	Addresses (decimal)
I	0.0 to 8191.7
O	0.0 to 8191.7
M	0.0 to 8191.7
SM	0.0 to 31.7
D	0.0 to 511.7
DX	0.0 to 511.7
DM	0.0 to 511.7
DF	0.0 to 32767.7
T-state	0 to 256
Z-state	0 to 256
P	0 to 62
DM	0 to 1023
PM	0 to 1023

6.11 Byte addresses

Operand	Address (decimal)	Comment
I	0 to 8191	
O	0 to 8191	
T-act. val. T-state	0 to 256 0 to 256	Timer range 10 ms to 1023 s; (Matrix: 0.01; 0.1; 1; 10 s)
Z-act. val. Z-state	0 to 256 0 to 256	Counter range: 0 to 8191
M	0 to 8191	
S	0 to 511	Managed values: ● System clock ● Error codes ● Times of time-controlled processing ● Versions, etc.
P	0 to 62	
DF	0 to 32767	
DM	0 to 511	
D	0 to 511	
DX	0 to 511	

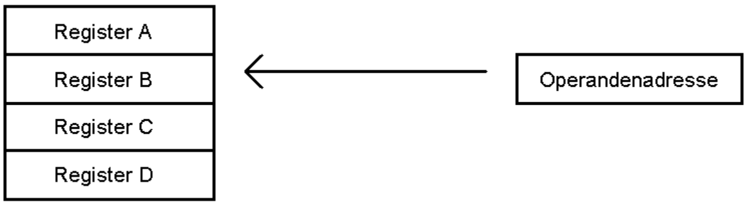
The even-numbered byte addresses are used as word addresses. For double word addresses the byte addresses have to be divisible by four.

6.12 Addressing modes

6.12.1 Absolute addressable operands

reading Byte, word, double word, REAL, LREAL	E, A, M, T, Z and P const., DF, DP, D, DX, SM, S	for T/C, actual values apply
writing: Byte,word,double word, REAL, LREAL	A, M, P, DF, DP, D, DX, S	P writing, depending on assigned operand

6.12.2 Direct addressing of all absolute addressable operands

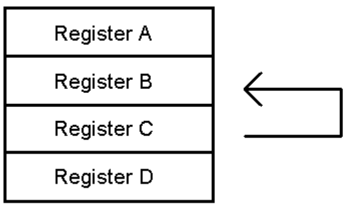


Examples:

```
L     B   E10,B                 ; Load the state of the
                                ; input book E10 in
                                ; the low byte of the
                                ; low word of B

L     W   100,C                 ; Load the value 100 in the
                                ; low word of the
                                ; register C
```

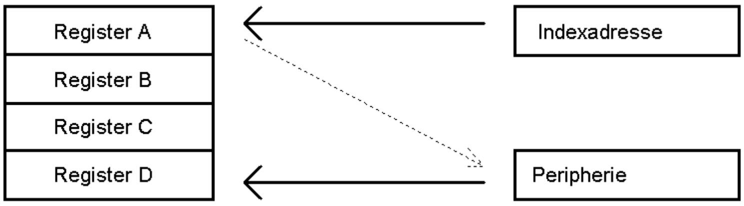
6.12.3 Register-to-register addressing



Example:

```
L  W  C,B     ; Load the contents of the low word
           ; of register C
           ; into the low word of
           ; register B
```

6.12.4 Register indirect addressing



Examples

```
L     D   10,A              ; Load indices as
                            ; byte number low word
                            ; from A, high word
                            ; is deleted

L     W   E[A],D            ; Load the state of E10
                            ; (address in A) into the
                            ; low word of register D
          ; load
```

☞ **When employed as index register 32 bits are always used.**

6.12.5 iPCL indirect addressing

Indirect addressing, whether word/byte or bit-oriented, is accomplished with the use of an operand prefix containing the operand identifier (operand ID) and operand address. This greatly facilitates the handling and monitoring of operand addresses.

In addition, all data and program modules can be called indirectly.

The operand prefix is structured as follows:

OPD[R] OPD = Operand ID
[R] = Operand address (index address) in the registers A, B, C, D

☞ **When loading index addresses into one of the registers, double word D must always be used as a supplement because the registers are 32 bits wide, and the HIGH word must be deleted!**

Principle of indirect addressing, using the example of a block transfer via program loop:

Task to be accomplished:

Transfer of 5 input words starting at address I10 into marker words from address M50 upward.

```
L      W   5,A            ; Load the loop counter
L      D   10,B           ; Load the base byte address I10
L      D   50,C           ; Load the base byte address M50
Continue:                 ; Loop entry label
L      W   E[B],D         ; Load contents
                          ; (Operand state)
T      W   D,M[C]         ; Write state that was loaded
INC    D   B,2            ; Next I-word (byte addr. + 2)
INC    D   C,2            ; Next M-word
DEC    D   A,1            ; Loop counter -1
SPN        continue       ; Not all words processed yet
```

Indirect byte addresses

OPD-ID	Byte address (dec.)	Instructions [Reg]	Examples
I	0 to 8191	L	L D 10,A
O	0 to 8191	L, T	L W OPD[A],B
T-act. val.	0 to 255	L	
Z-act. val.	0 to 255	L	L D 10,A
M	0 to 8191	L, T	T W B,OPD[A]
P	0 to 62	L	
S	0 to 511	L, T	
SM	0 to 31	L, T	
DF	0 to 32767	L, T	
DM	0 to 511	L, T	
D	0 to 511	L, T	
DX	0 to 511	L, T	

In order to address the next byte or next T/C the address needs to be incremented by 1. In order to address the next word the address needs to be incremented by 2.

Indirect bit addresses

OPD-ID	Byte address (dec.)	Instructions	For examples of OPD see column 1
I	0 to 65455	A, AN, O, ON	
O	0 to 65455	A, AN, O, ON, S, R, =	L D 10,AU OPD[A]
M	0 to 65455	A, AN, O, ON, S, R, =	= OPD[A]
S	0 to 4095	A, AN, O, ON	
SM	0 to 255	A, AN, O, ON	
D	0 to 4095	A, AN, O, ON, S, R, =	
DX	0 to 4095	A, AN, O, ON, S, R, =	
DM	0 to 4095	A, AN, O, ON, S, R, =	
DF	0 to 262143	A, AN, O, ON, S, R, =	
T-state	0 to 255	A, AN, O, ON	
Z-state	0 to 255	A, AN, O, ON	

To address the next bit relative to a given starting address, this address must be incremented by 1.

Indirect module addresses

Operand	Module number	Instructions [Reg]	Example
DM	0 to 1023	CMx BXx	L D 10,A CM DM[A]
PM	0 to 1023	CMx CMx	L D 100,A CM PM[A]

To address the next module relative to a given module number, it must be incremented by 1.

In the case of a range violation or if the module is not available, the controller will enter STOP mode. In both instances, the cause of the error can be indicated by the Programming Unit (PG).

6.13 Parameter transfer

When a program module is called up, up to 63 parameters can be transferred. The number of parameters transferred is specified in the module call-up command. Then the parameters follow, starting with P0. In a PM that has been called, these parameters can also be processed indirectly: (L D P[R],R).

☞ **The indirect processing of parameters is only possible from WinSPS version 3.0 onwards.**

The applicable operand attributes are listed below:

- D – double word (default)
- W – word
- B – byte

Bit operands are programmed without the use of attributes.

☞ **Timers and counters are transferred without operand attributes to facilitate their use as both word (i.e. timer / counter values) and as bit (i.e. timer / counter status) in the module to be called.**

Example: Parameter transfer

```
CM        PM100,7         ; Call PM100 using 7 parameters
P0    D   43              ;Parameter P0: PM no. as constant 43
P1    D   4               ;Parameter P1: DM no. as constant K4
P2    W   O56             ;Parameter P2: Ouput word at byte addr. O56
P3        I7.3            ;Parameter P3: Input bit I7.3
P4        T2              ;Parameter P4: Timer T2
P5        C13             ;Parameter P5: Counter C13
P6        O10.0           ;Parameter P6: Output bit O10.0

          Utilization of parameters in called-up module PM100:

L     D   P1,A            ;Load data module no. 4
CM        DM[A]           ;Open DM4
BX        -DB5

L     D   P0,A            ;Load PM no. 43
CM        PB[A]0.2        ;Use 2 parameters to call PM43
P0    D   43              ;Parameter P0: D2  of active 1st DM (DM4)
P1    D   4               ;Parameter P1: DX6 of active 2nd DM (DM5)

L     W   P2,A            ;Load output word O56

L     W   P4,B            ;Load timer value from T2 to B

U         P3              ;E7.3
A         P4              ;Status of T2
A         P5              ;Status of C13
=         P6              ;O10.0
```

6.14 Addressing limits

Direct addressing

In direct addressing, addressing limits are determined by the operand attribute.

Byte	Address as desired
Word	Address even-numbered
Double word	Address divisible by 4
REAL	Address divisible by 4
LREAL	Address divisible by 8

Example:

Operand	B	W	D	R	L
M0	x	x	x	x	x
M1	x				
M2	x	x			
M3	x				
M4	x	x	x	x	
M5	x				
M6	x	x			
M7	x				
M8	x	x	x	x	x

Indirect addressing

Example:

```
L   D   0,A       ;Address byte 0
L   D   M[A],B    ;State of  M0+M1+M2+M3 is read
INC D   A,1       ;Address byte 1
L   D   M[A],B    ;State of  M1+M2+M3+M4 is read
INC D   A,1       ;Address byte 2
L   D   M[A],B    ;State of  M2+M3+M4+M5 is read
INC D   A,1       ;Address byte 3
L   D   M[A],B    ;State of  M3+M4+M5+M6 is read
INC D   A,1       ;Address byte 4
L   D   M[A],B    ;State of  M4+M5+M6+M7 is read
```

Parameterized addressing

Parameterized addressing is not subject to the same addressing limits as indirect addressing.

Example:

Parameter definition	Parameter query	Reads the following:
P0 M1	L B P0,A	M1
P1 M3	L W P1,A	M3 and M4
P2 M5	L D P2,A	M5 to M8
P3 M7	L D P3,A	M7 to M10
P1 M11	L L P1,A	M11 to M18

Notes:

7 Instruction set

7.1 Structure of controller instructions

Controller instruction					Line comment
Operations part	Operand attribute	Source operand	Destinatio n operand		
OPP	OPA	SRC	, DEST	;	Instruction description

Examples:

```
U            I0.0
U      W    -Name  ,   O
L      B     O0    ,   B
T      D     C     ,  M12
MUL    W    1234   ,   D
```

7.2 Flags

The flags are influenced by the following instruction groups:

- Bit instructions
- Compare
- Convert
- Swap
- Increment
- Decrement
- Shift
- Rotate
- Add
- Subtract
- Multiply
- Divide

They can be used not only in program processing instructions (jumps, module instructions) but also in logical links (special marker queries).

Flags	Display in WinSPS	JP... CM...	Flag query	Description
CY=1 CY=0	C	...C ...CN	U CY AN CY	Carry Carry Not
O=1 O=0	O	...O ...ON	U O AN O	OverflowOverflow not
Z=1 Z=0	Z	...Z ...N	U Z AN Z	Zero Not Zero
N=1 N=0	N	...M ...P	U N AN N	Negative/minus Positive
AG=1		...AG	No flag link	Arithmetical greater
AG=0	N v Z	...MZ	U Z O N AN O ON N U O	Minus / zero
LG=1		...LG	AN Z AN CY	Logical greater
LG=0	C v Z	...CZ	U Z O CY	Carry/zero

7.3 Key to abbreviations

OPP		Operation
OPA		Operand attribute
	B	Byte
	W	Word
	D	Double word
	R	REAL
	L	LREAL (LongReal)
SRC		Source operand
DEST		Destination operand
	I	Input
	O	Output
	M	Markers
	K	Constants
	SM	Special marker
	T	Timer
	Z	Counters
	D	Data word (within data modules)
	DM	Data buffer
	DF	Data field
	S	System area
	DM	Data module
	DX	2. 2nd active data module
	PM	Program module
	SYM	Symbolic
	R.bit	Register bit with R = A, B, C, D, and bit = 0 to 31
	OPD[R]	Register indirect with operand prefix
	TI	Time interrupt (time-controlled processing)
RG		Program branch
	A	Operation permitted at RG beginning
	E	Operation concluding RG
Addr.		Addressing mode
	D	Direct
	R	Register A, B, C, or D
	[R]	Register indirect with operand prefix
Flag		State bit
	V	Link result RES
	CY	Carry
	O	Overflow
	Z	Zero
	N	Negative

7.4 Bit instructions

Bit instructions modify the state bits C, Z, O, and N.

☞ **Exception: Flags themselves are not changed by a binary flag query.**

Links are interpreted in accordance with the Boolean “AND” before “OR” logic principle. Parenthesized instructions are used to form logical intermediate results.

Controller instruction				RG		Addr. type			Flag					Example		Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z			
U		I/O/M/SM		●		●			●	●	●		●	U	I0.0	AND link, query status 1
		T/C/SYM		●		●			●	●	●		●	U	T0	
		R.bit		●			●		●	●	●		●	U	A.0	
		OPD[R]		●				●	●	●	●		●	U	M[A]	
		P		●		●			●	●	●		●	U	P0	
		S/D/DX/DF/DP		●		●			●	●	●		●	U	D0.0	
		CY/Z/O/N		●		●								U	CY	
AN		I/O/M/SM		●		●			●	●	●		●	AN	A0.0	AND link, query status 0
		T/C/SYM		●		●			●	●	●		●	AN	Z0	
		R.bit		●			●		●	●	●		●	AN	B0.0	
		OPD[R]		●				●	●	●	●		●	AN	M[B]	
		P		●		●			●	●	●		●	AN	P1	
		S/D/DX/DF/DP		●		●			●	●	●		●	AN	D0.0	
		CY/Z/O/N		●		●								AN	CY	
O		I/O/M/SM		●		●			●	●	●		●	O	M0.0	OR link, query status 1
		T/C/SYM		●		●			●	●	●		●	O	-SYMBOL	
		R.bit		●			●		●	●	●		●	O	C0.0	
		OPD[R]		●				●	●	●	●		●	O	MD[C]	
		P		●		●			●	●	●		●	O	P10	
		S/D/DX/DF/DP		●		●			●	●	●		●	O	D0.0	
		CY/Z/O/N		●		●								O	CY	
ON		I/O/M/SM		●		●			●	●	●		●	ON	SM31.7	OR link, query status 0
		T/C/SYM		●		●			●	●	●		●	ON	Name	
		R.bit		●			●		●	●	●		●	ON	D.0	
		OPD[R]		●				●	●	●	●		●	ON	M[D]	
		P		●		●			●	●	●		●	ON	P62	
		S/D/DX/DF/DP		●		●			●	●	●		●	ON	D0.0	
		CY/Z/O/N		●		●								ON	CY	
=		A/M/SYM			●	●			●	●	●		●	=	A0.0	Assign result when RES = 1
		S/D/DX/DF/DP			●	●			●	●	●		●	=	D0.0	
		P			●	●	●		●	●	●		●	=	P0	
		OPD[R]			●			●	●	●	●		●	=	M[A]	
		R.bit			●		●		●	●	●		●	=	A.0	
S		A/M/SYM			●	●			●	●	●		●	S	M0.0	Set bit HIGH when RES = 1
		S/D/DX/DF/DP			●	●			●	●	●		●	S	D0.0	
		P			●	●	●		●	●	●		●	S	P1	
		OPD[R]			●			●	●	●	●		●	S	M[B]	
		R.bit			●		●		●	●	●		●	S	B0.0	

OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
R		A/M/SYM			●	●			●	●	●		●	R -SYMBOL	Set bit LOW when RES = 1
		S/D/DX/DF/DP			●	●			●	●	●		●	R D0.0	
		P			●	●	●		●	●	●		●	R P62	
		OPD[R]			●			●	●	●	●		●	R M[C]	
		R.bit			●		●		●	●	●		●	R C0.0	
P		R.bit					●		●	●	●		●	P A.0	Check register bit for status = 1 if met: C = 1
PN		R.bit					●		●	●	●		●	P A.15	Check register bit for status = 0 if met: C = 1
(									●	●	●		●	(	AND opening bracket
)									●	●	●		●	)	Closing bracket
O(									●	●	●		●	O(	OR opening bracket
)N									●	●	●		●	)N	Negation of bracket contents

7.5 Timer programming

The iPCL provides 256 timer circuits, T0 through T255.

These can be used in the following modes:

- SP Pulse
- SPE Start pulse extended
- SR Start time as raising delay
- SF Start time as falling delay
- SRE Start time as raising delay extended

Starting non-residual timers

Starting the non-residual starting timers SP, SPE, SR and SRE requires a positive transition of the timer start condition. However, they are also started if at the time of first addressing (1st PLC cycle) after startup or restart the start condition equals 1.

Residual timers

In the case of residual timers, the flank marker is retained, i.e. whether a 1 will start the timer at the time of first addressing (1st PLC cycle) after startup or restart, depends on the start condition prior to STOP or Power-OFF.

Off-delay

In the case of the start time as falling delay, a “0” will not start the timer during the initial processing. Predefining the timer start condition with 1 is possible as early as in the startup OM, provided that the information about residual characteristics (see Section on Residual characteristics) is considered.

The timers are decremented in the I/O state. A timeout is thus recognized only in the I/O state, and not during the program cycle!

Because a timer is decremented in the I/O state by a multiple of the declared time matrix, it is useful to select a time matrix that is a small as possible.

The timer starts immediately upon a positive transition of the timer start condition.

7.5.1 Timer instructions

Timer starts are activated only when the RES signal undergoes a transition from 0 ↑1. In advance of the timer start, the time value is loaded into the register being used. Reset and stop functions of timers are always RES signal-dependent. The timer status for logical links is instruction-dependent, and may be taken from the timer diagrams.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SP		R	, Tn		●	●								SP A,T0	Pulse
			, SYM			●								SP A,-Symbol	
			, T[R]					●						SP A,T[B]	
			, P			●								SP A,P0	
SPE		R	, Tn		●	●								SPE A,T0	Start pulse extended
			, SYM			●								SPE A,-Symbol	
			, T[R]					●						SPE A,T[B]	
			, P			●								SPE A,P0	
SR		R	, Tn		●	●								SR A,T0	On-delay
			, SYM			●								SR A,-Symbol	
			, T[R]					●						SR A,T[B]	
			, P			●								SR A,P0	
SF		R	, Tn		●	●								SF A,T0	Off-delay
			, SYM			●								SF A,-Symbol	
			, T[R]					●						SF A,T[B]	
			, P			●								SF A,P0	
SRE		R	, Tn		●	●								SRE A,T0	Start time as raising delay extended
			, SYM			●								SRE A,-Symbol	
			, T[R]					●						SRE A,T[B]	
			, P			●								SRE A,P0	
RT		Tn			●	●								RT T0	Set timer LOW when RES = 1
		SYM				●								RT -Symbol	
		T[R]						●						RT T[B]	
		P				●								RT P0	
TH		Tn			●	●								TH T0	Timer STOP when RES = 1, timer continues when RES = 0
		SYM				●								TH -Symbol	
		T[R]						●						TH T[B]	
		P				●								TH P0	

7.5.2 Time format

The following applies to the time format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	R	R	W	W	W	W	W	W	W	W	W	W
				Time-matrix		Timer value 1 – 1023									

0	0	0: 10 ms		
0	1	1: 100 ms	Program entry of time constants:	
1	0	2: 1 s	w.r	with time value w = 1 –1023
1	1	3: 10 s		and time matrix r = 0 – 3

Example:

Timer T100 is to be started at 15 sec:

```
L      W    T#15s,A      ;15s declaration in the CL500 1s time matrix
U      B    -start
SPE         A,T100
```

Same function but with smaller time matrix, i.e. higher accuracy:

```
L      W    T#15000ms,A  ;15s declaration in the CL500 100ms time
matrix
U      B    -start
SPE         A,T100
```

Timer start with the assistance of the PG time matrix:

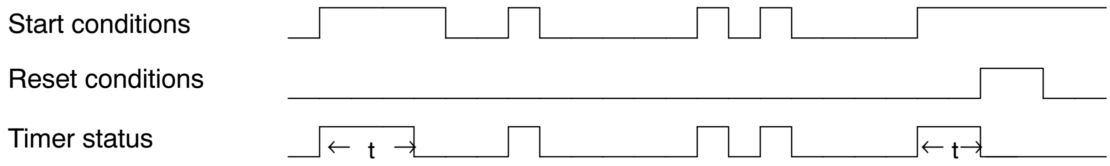
```
L      W    T#15.2,A     ;15s declaration in the PG 1s time matrix
U      B    -start
SPE         A,T100
```

Same function but with smaller time matrix, i.e. higher accuracy:

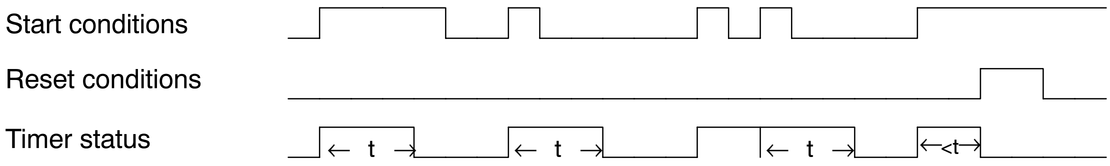
```
L      W    T#150.1,A    ;15s declaration in the PG 100ms time matrix
U      B    -start
SPE         A,T100
```

7.5.3 Timer diagrams

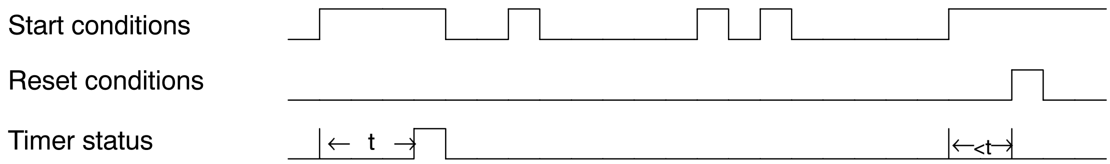
SP – Start time as pulse



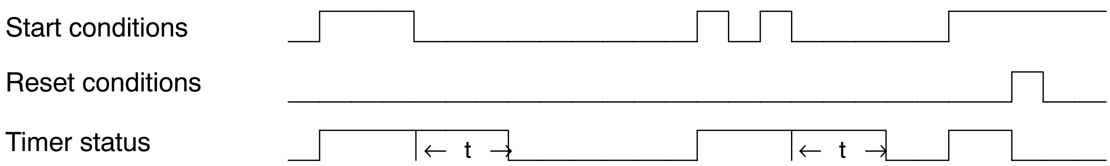
SPE – Start pulse extended



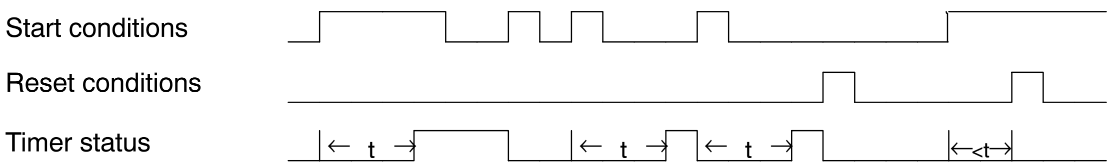
SR – Start time as raising delay



SF – Start time as falling delay



SRE – Start time as raising delay extended



7.6 Counter instructions

The setting of counters and counting up and down occurs only when the RES signal undergoes a transition from 0 -> 1.

In advance of the setting, the required counter content is loaded into the register being used.

Counter resetting always occurs static RES signal-dependent.

The counter status for logical links depends on the counter content.

- For counter values > 0 the status = 1
- For value = 0 status = 0

The counting range is between 0 to 8191.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
Zn		R	, Zn		●	●								SC A,Z0	Set counter HIGH
			, SYM			●								SC A,-Symbol	
			, Z[R]					●						SC A,Z[B]	
			, P			●								SC A,P0	
CU		Zn			●	●								CU Z0	Count up
		SYM				●								CU -Symbol	
		Z[R]						●						CU Z[B]	
		P				●								CU P0	
CD		Zn			●	●								CD Z0	Count down
		SYM				●								CD -Symbol	
		Z[R]						●						CD Z[B]	
		P				●								CD P0	
RC		Zn			●	●								RC Z0	Set counter LOW when RES = 1
		SYM				●								RC -Symbol	
		Z[R]						●						RC Z[B]	
		P				●								RC P0	

7.7 Digital links

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
U	B	I/O/M/SM	, R			•				0	0	•	•	U B E0,A	Digital AND link
	W	T/C/K/SYM				•				0	0	•	•	U W T0,B	between source and destina-
	D	S/DF/DP				•				0	0	•	•	U B S0,C	tion.
		D/DX				•				0	0	•	•	U W D0,D	The result is written to destina-
															tion.
		R					•			0	0	•	•	U W A,B	
		OPD[R]						•		0	0	•	•	U B M[B],C	
		P				•				0	0	•	•	U W P0,D	
AN	B	I/O/M/SM	, R			•				0	0	•	•	AN B E127,A	Digital AND NOT
	W	T/C/K/SYM				•				0	0	•	•	AN W T127,B	link between
	D	S/DF/DP				•				0	0	•	•	AN B S511,C	source and destination.
		D/DX				•				0	0	•	•	AN W D510,D	The result is written to destina-
		R					•			0	0	•	•	AN W A,B	tion.
		OPD[R]						•		0	0	•	•	AN B M[B],C	
		P				•				0	0	•	•	AN W P62,D	
O	B	I/O/M/SM	, R			•				0	0	•	•	O B E0,A	Digital OR link
	W	T/C/K/SYM				•				0	0	•	•	O W T0,B	between source and
	D	S/DF/DP				•				0	0	•	•	O B S0,C	destination.
		D/DX				•				0	0	•	•	O W D0,D	The result is written to destina-
		R					•			0	0	•	•	O W A,B	tion.
		OPD[R]						•		0	0	•	•	O B M[B],C	
		P				•				0	0	•	•	O W P0,D	
ON	B	I/O/M/SM	, R			•				0	0	•	•	ON B E127,A	Digital OR NOT
	W	T/C/K/SYM				•				0	0	•	•	ON W T127,B	link between
	D	S/DF/DP				•				0	0	•	•	ON B S511,C	source and destination.
		D/DX				•				0	0	•	•	ON W D510,D	The result is written to destina-
		R					•			0	0	•	•	ON W A,B	tion.
		OPD[R]						•		0	0	•	•	ON B M[B],C	
		P				•				0	0	•	•	ON W P62,D	
XO	B	I/O/M/SM	, R			•				0	0	•	•	XO B E0,A	EXCLUSIVE OR
	W	T/C/K/SYM				•				0	0	•	•	XO W T0,B	link between
	D	S/DF/DP				•				0	0	•	•	XO B S0,C	source and destination.
		D/DX				•				0	0	•	•	XO W D0,D	The result is written to destina-
		R					•			0	0	•	•	XO W A,B	tion.
		OPD[R]						•		0	0	•	•	XO B M[B],C	
		P				•				0	0	•	•	XO W P0,D	
XON	B	I/O/M/SM	, R			•				0	0	•	•	XON B E127,A	EXCLUSIVE OR NOT
	W	T/C/K/SYM				•				0	0	•	•	XON W T127,B	link between
	D	S/DP/DF/DP				•				0	0	•	•	XON B S511,C	source and destination.
		D/DX				•				0	0	•	•	XON W D510,D	The result is written to destina-
		R					•			0	0	•	•	XON W A,B	tion.
		OPD[R]						•		0	0	•	•	XON B M[B],C	
		P				•				0	0	•	•	XON W P62,D	

7.8 SWAP instructions

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SWAP															Change in register
	W	R					•							SWAP W O	High byte « Low byte
	D													SWAP D O	High word « Low word

7.9 Compare instruction

The universally applicable CPLA (Compare Logical and Arithmetical) instruction is available for Compare operations. This facilitates both logical and arithmetic compare operations.

For reasons of compatibility the purely logical CPL instruction was also implemented; it is used to map binary result queries also in special markers.

The logical compare operation regards the bytes, words, or double words to be compared as unsigned integers, i.e. as unsigned 8, unsigned 16, or unsigned 32.

The arithmetical compare operation regards the bytes, words, or double words to be compared as signed integers, i.e., as signed 8, signed 16, or signed 32.

After a compare instruction the flags indicate the result of the compare.

Controller instruction				RG		Addr.			Flag							Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z	AG	LG		
CPLA	B	I/O/M/SM	, R			•	•	•		•	•	•	•	•	•	CPLA W E62,A	Arithmetical compare function.
	W	T/C/K/SYM				•				•	•	•	•	•	•	CPLA B 255,B	The result may be used for log-
	D	S/DF				•				•	•	•	•	•	•	CPLA W DF510,C	ical and arithmetical purposes.
		D/DX/DP				•				•	•	•	•	•	•	CPLA B D511,D	
		R				•				•	•	•	•	•	•	CPLA B B,C	Compare vlues
		OPD[R]								•	•	•	•	•	•	CPLA W M[C],D	Logical: positive, integer
		P								•	•	•	•	•	•	CPLA B P62,A	arithm.: two's complement,
																	signed
CPL	B	I/O/M/SM	, R			•	•	•		•	•	•	•	•	•	CPL W E62,A	Logical compare operation.
	W	T/C/K/SYM				•				•	•	•	•	•	•	CPL B 255,B	The result may be used for log-
	D	S/DF				•				•	•	•	•	•	•	CPL W DF510,C	ical purposes only,
		D/DX/DP				•				•	•	•	•	•	•	CPL B D511,D	i.e. the values will be treated
		R				•				•	•	•	•	•	•	CPL B B,C	as positive integers.
		OPD[R]								•	•	•	•	•	•	CPL W M[C],D	
		P								•	•	•	•	•	•	CPL B P62,A	

CPLA compare vlues:

- Logical: positive, integer
- Arithmetical: two's complement, signed integer

After a compare operation, the flags or special markers provide information about the result of the compare.

Examples:

Compare destination (A) with source (B)		CPL B,A Logical		CPLA B,A Logical		CPLA B,A Arithmetical
		Jump instruction	Flag query	Jump instruction	Flag query	Jump instruction
Equal	A = B	JPZ	U SM31.7	JPZ	U Z	JPZ
Unequal	A ≠ B	JPN	UN SM31.7	JPN	UN Z	JPN
Less than	A < B	JPN	U SM31.6	JPCY	U CY	JPM
Less than / equal	A ≤ B	JPCZ	UN SM31.0	JPCZ	U Z O CY	JPMZ
Greater than	A > B	JPLG	U SM31.0	JPLG	UN CY UN Z	JPAG
Greater than / equal	A ≥ B	JPCN	UN SM31.6	JPCN	UN CY	SPP

☞ **When using the CPLA instruction, the evaluation of the compare results must always be programmed immediately following the compare instruction itself. The user is advised to bear in mind that with the exception of flag queries, binary operations will cause a modification of the flags. Therefore, a compare result can be used only in a link. Following this, another CPLA instruction must again be programmed.**

☞ **The special markers that are influenced only by the CPL instruction will remain unaffected until the next CPL instruction.**

7.10 Load instructions

Load instructions (L) are used to write statuses or values from operands into registers. Signal statuses of inputs / outputs are loaded from the periphery image.

In the event that the status of inputs or outputs is to be loaded directly from the peripherals during the program cycle, then this status must be loaded into the image (LD) before the actual load instruction (L) is issued.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
L	B	I/O/M/SM	, R			•								L W E0,A	Load content of SRC
	W	T/C/K/SYM				•								L B 0,B	into DEST. (Read I/O image)
	D	DF/DP				•								L W DF0,C	Read value
		D/DX				•								L B D0,D	
		R					•							L B B,C	
		OPD[R]						•						L W M[C],D	
		P				•								L B P0,A	
		P[R]						•						L D P[A],B	
		OM, FC,				•								L DB10,A	Load DB10 module size
		DM													
LD		I	, K			•								LD E0,20	Load 20 bytes* of input sta- tuses into image, starting with I0.
		I[R]	, [R]					•						LD E0,[B]	Load I-statuses into image, starting with I0; byte* count in B.
														LD E[A],[B]	Load I-statuses into image (start address in A), (byte* count in B). * max. byte count = 256

Example of direct loading:

```
LD      D     I12,4          ; Load byte from I12 from the bus master into I-image

L       D     I12,4          ; Load statuses I12 through I15 into register A
```

When using the ”indirect parameter” load instruction (L D P[R],R), the WinSPS is unable to perform a syntax check because it cannot foresee which operand address will actually be addressed by the parameter. The controller may enter STOP mode. The user is therefore advised to ensure the required syntax for this instruction is used.

7.11 Tranfer instructions

Transfer instructions (T) are used to write statuses or values from registers to operands. Signal statuses from outputs are written into the periphery image. In the I/O state this image is then transferred to the outputs.

In the event that the statuses of outputs are to be sent directly to the peripherals during the program cycle, then the transfer instruction (TD) will be used.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
T	B	R	, A/M/SYM			•								T W A,M0	Transfer content of SRC
	W		, S/DF			•								T B B,DF0	to DEST. (Write I/O image)
	D		, D/DX/DP			•								T W C,D0	Write value.
			, R				•							T W A,B	
			, OPD[R]					•						T B B,M[C]	
			, P			•								T W D,P0	
TD		O	, K			•								TD O0,20	Send 20 bytes* of output statuses from image to outputs, starting with O0.
		O[R]	, [R]					•						TD O0,[B]	Send O-statuses from image to outputs, starting with O0. (Byte count* in B).
														TD A[A],[B]	Send O-statuses from image to outputs (start address in A). (Byte count* in B).
															* max. byte count = 256

Example of direct transfer:

```
L     D    16#1234FFFF,A ; Load hex constant into register A
T     D    A,A12         ; and write to O-image.
TD    D    A12,4         ;transfer 4 bytes into bus master
                         ;for A12-A15.
```

7.12 Convert instructions

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
BID	B W D	R					•			0	•	0	•	BID W O BID B B	Binary → BCD (decimal) result > 9999 sets the overflow bit
DEB	B W D	R					•			0	•	0	•	DEB W C DEB B D	BCD (decimal) → Binary wrong BCD coding sets the overflow bit.
CMP	B W D	R					•			•	•	•	•	CMP W O CMP B B	Converts register contents to the two's complement.
N	B W D	R					•			0	•	0	•	N W C N B D	Negates register contents, one's complement.

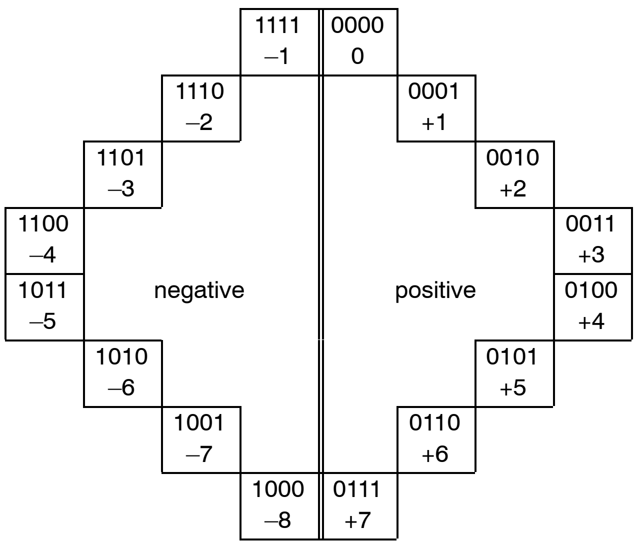
Positive and negative numbers are differentiated by the status of the MSB.

OPA	Positive	Range	Negative	Range
Double word	Bit 31 = 0	0 to +2,147,483,647	Bit 31 = 1	0 to –2,147,483,648
Word	Bit 15 = 0	0 to +32,767	Bit 15 = 1	0 to –32,768
Byte	Bit 7 = 0	0 to +127	Bit 7 = 1	0 to –128

Example: The representation of positive and negative numbers.

By way of illustration, a 4-bit number (nibble) is used here; the nibble data format is not supported by the controller.

```
  0 1 1 0   positive number 6
  1 0 0 1   negation one's complement
+       1
  -------
  1 0 1 0   two's complement = negative number 6
```



7.13 Increment & Decrement instructions

Increment / decrement the contents of source operand SRC:

- by the number n; (where n=1 to 127)
- when n = 0, and when [C], by the number stored in C, max. 127.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
INC	B W D	R	, n , 0 , [C]			•	•	•		• • •	• • •	• • •	• • •	INC BY A,5 INC W A,0 INC W B,[C]	Increment the contents of the SRC
DEC	B W D	R	, n , 0 , [C]			•	•	•		• • •	• • •	• • •	• • •	DEC B A,5 DEC W A,0 DEC W B,[C]	Decrement the contents of the SRC

7.14 Stack instructions

The available stack size comprises 256 double words. In the event of underflow, special marker S28.4 in the system area goes HIGH; overflow sets the S28.5 to HIGH. The I/O state deletes the entire application stack.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
PUSH	D	R					•							PUSH D O	Saves the register contents to application stack, and lowers the stack address.
POP	D	R					•							POP D B	Raises the application stack address, and reads the saved contents from the stack.

7.15 No operation instructions & CARRY manipulations

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
NOP														NOP	No operation
SCY										•				SCY	Unconditionally set CARRY bit to 1.
RCY										•				RCY	Unconditionally set CARRY bit to 0.

7.16 Shift instructions

Shift the contents of source operand SRC:

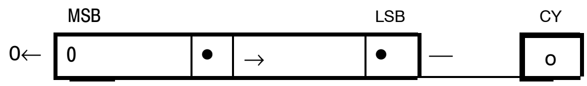
- by the number n
- when n = 0, and when [C], by the number stored in C.

  When OPA = D then n = 1 to 31
  When OPA = W then n = 1 to 15
  When OPA = B then n = 1 to 7

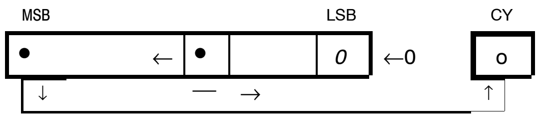
Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SLR	B	R	, n			•	•	•		•	•	•	•	SLR W A,7	SHIFT logical RIGHT
	W		, 0							•	•	•	•	SLR B B,[C]	
	D		, [C]							•	•	•	•		
SLL	B	R	, n			•	•	•		•	•	•	•	SLL W A,7	SHIFT logical LEFT
	W		, 0							•	•	•	•	SLL B B,[C]	
	D		, [C]							•	•	•	•		
SAR	B	R	, n			•	•	•		•	•	•	•	SAR W A,7	SHIFT arithmetical RIGHT
	W		, 0							•	•	•	•	SAR B B,[C]	
	D		, [C]							•	•	•	•		

Logical SHIFT:

SLR B , n



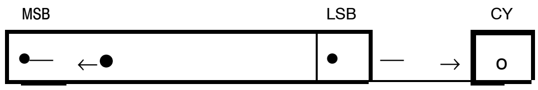
SLL B , n



Arithmetical SHIFT:

All bits being vacated are filled up with the contents of the MSB.

SAR B , n



In the case of shift operations exceeding one space the overflow bit is set HIGH after a ”1” was shifted through CY.

7.17 Rotate instructions

Shift the contents of source operand SRC:

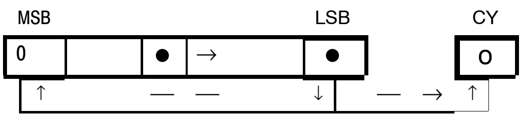
- by the number n
- when n = 0, and when [C], by the number stored in C.

  When OPA = D then n = 1 to 31
  When OPA = W then n = 1 to 15
  When OPA = B then n = 1 to 7

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
ROR	B	R	, n			•	•	•		•	•	•	•	ROR B A,7	Rotate RIGHT
	W		, 0							•	•	•	•	ROR W A,0	
	D		, [C]							•	•	•	•	ROR W B,[C]	
ROL	B	R	, n			•	•	•		•	•	•	•	ROL B A,7	Rotate LEFT
	W		, 0							•	•	•	•	ROL W A,0	
	D		, [C]							•	•	•	•	ROL W B,[C]	
RCR	B	R	, n			•	•	•		•	•	•	•	RCR B A,7	Rotate RIGHT through CARRY
	W		, 0							•	•	•	•	RCR W A,0	
	D		[C]							•	•	•	•	RCR W B,[C]	
RCL	B	R	, n			•	•	•		•	•	•	•	RCL B A,7	Rotate LEFT through CARRY
	W		, 0							•	•	•	•	RCL W A,0	
	D		, [C]							•	•	•	•	RCL W B,[C]	

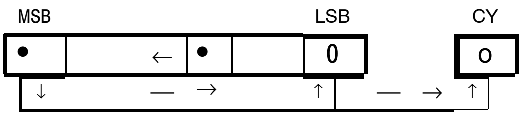
Rotate right:

ROR B , n



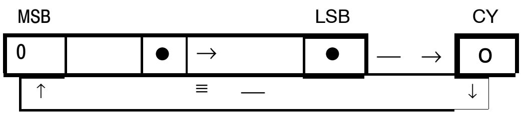
Rotate left

ROL B , n



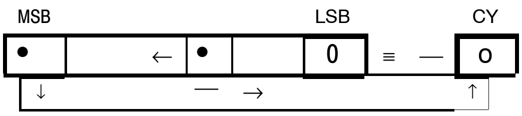
Rotate right through CARRY

RCR B , n



Rotate left through CARRY

RCL B , n



In the case of a rotation by more than one space, the following applies:

- The overflow bit goes HIGH when a 1 has been rotated through CY.
- The negative bit goes HIGH when the MSB contains a 1.

  MSB: Bit 7 when OPA = B
  Bit 15 when OPA = W
  Bit 31 when OPA = D

7.18 Fixed point arithmetic

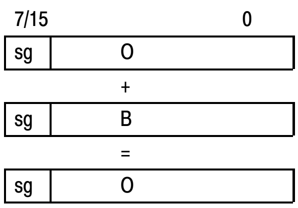
7.18.1 Add instructions

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
ADD	B	I/O/M/SM	, R			•				•	•	•	•	ADD W E0,A	Fixed point addition of signed integers:
	W	T/C/K/SYM				•				•	•	•	•	ADD B 0,B	SRC + DEST = DEST
	D	S/DF/DP				•				•	•	•	•	ADD W DP0,C	
		D/DX				•				•	•	•	•	ADD B D0,D	
		R					•			•	•	•	•	ADD B B,C	
		OPD[R]						•		•	•	•	•	ADD W M[C],D	
		P				•				•	•	•	•	ADD B P0,A	
ADC	B	I/O/M/SM	, R			•				•	•	•	•	ADC W E0,A	Fixed point addition of signed integers allowing for carry (CY):
	W	T/C/K/SYM				•				•	•	•	•	ADC B 0,B	SRC + DEST + CY = DEST
	D	S/DF/DP				•				•	•	•	•	ADC W DP0,C	
		D/DX				•				•	•	•	•	ADC B D0,D	
		R					•			•	•	•	•	ADC B B,C	
		OPD[R]						•		•	•	•	•	ADC W M[C],D	
		P				•				•	•	•	•	ADC B P0,A	

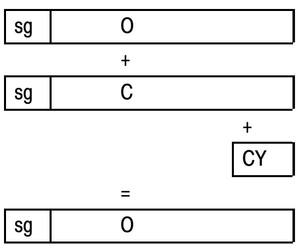
Byte, word, and double-word addition

MSB:
- Bit 7 when OPA = B
- Bit 15 when OPA = W
- Bit 31 when OPA = D

ADD OPA B , O



ADC OPA C , O

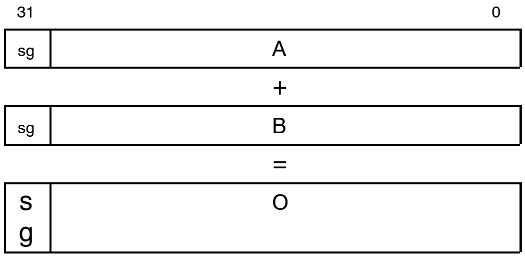


Quad-word addition: Value 1 + value 2

Value 1: LOW DW in B, HIGH DW in A
Value 2: LOW DW in D, HIGH DW in C

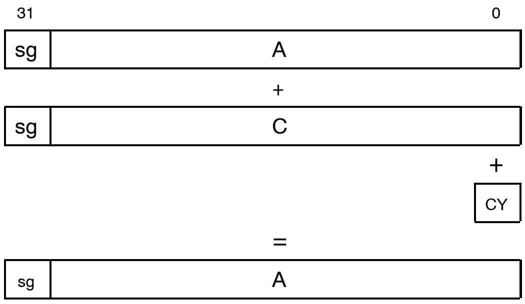
Low-DW

ADD D B , O



High-DW:

ADD D B , O



7.18.2 Substract instructions

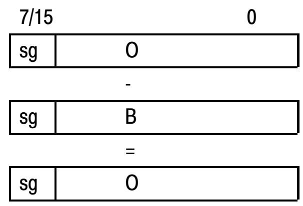
Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SUB	B	I/O/M/SM	, R			•				•	•	•	•	SUB W E0,A	Fixed point subtraction of
	W	T/C/K/SYM				•				•	•	•	•	SUB B 0,B	signed integers
	D	S/DP/DF/DP				•				•	•	•	•	SUB W DP0,C	DEST - SRC = DEST.
		D/DX				•				•	•	•	•	SUB B D0,D	
		R					•			•	•	•	•	SUB B B,C	
		OPD[R]						•		•	•	•	•	SUB W M[C],D	
		P				•				•	•	•	•	SUB B P0,A	
SBB	B	I/O/M/SM	, R			•				•	•	•	•	SBB W E0,A	Fixed point subtraction of
	W	T/C/K/SYM				•				•	•	•	•	SBB B 0,B	signed integers allowing for
	D	S/DP/DF/DP				•				•	•	•	•	SBB W DP0,C	negative carry
		D/DX				•				•	•	•	•	SBB B D0,D	(– Carry = Borrow)
		R					•			•	•	•	•	SBB B B,C	DEST - SRC - CY = DEST.
		OPD[R]						•		•	•	•	•	SBB W M[C],D	
		P				•				•	•	•	•	SBB B P0,A	

Byte, word, double word subtraction:

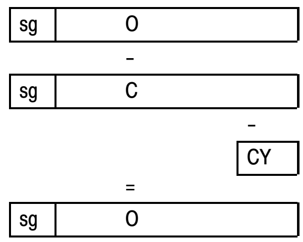
MSB:

- Bit 7 when OPA = B
- Bit 15 when OPA = W
- Bit 31 when OPA = D

SUB OPA B , O



SBB OPA C , O

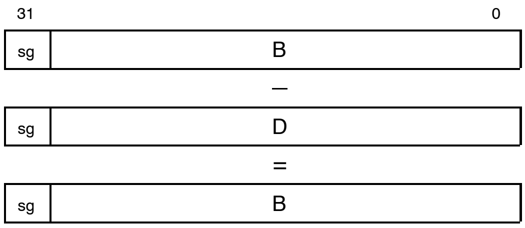


Quad-word subtraction: Value 1 – value 2

Value 1: LOW DW in B, HIGH DW in A
Value 2: LOW DW in D, HIGH DW in C

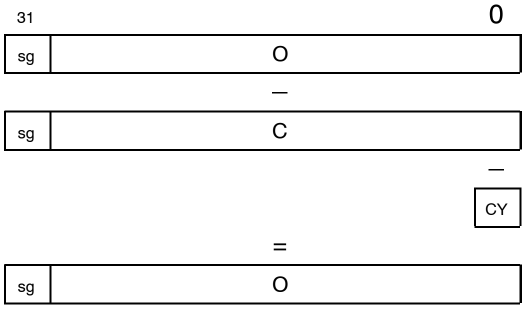
Low-DW

SUB D D , B



High-DW:

SBB W C , O



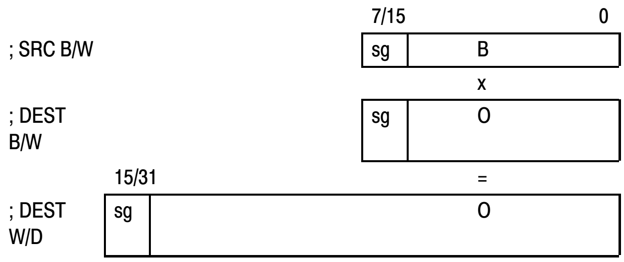
7.18.3 Multiply instructions

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
MUL	B W D	K R	, R			•	 •			0 0	0 0	• •	• •	MUL B 100,A MUL W B,A MUL D B,A	Fixed-point multiplication of signed integers.

In multiplication, the product always occupies the double width of the output operands.

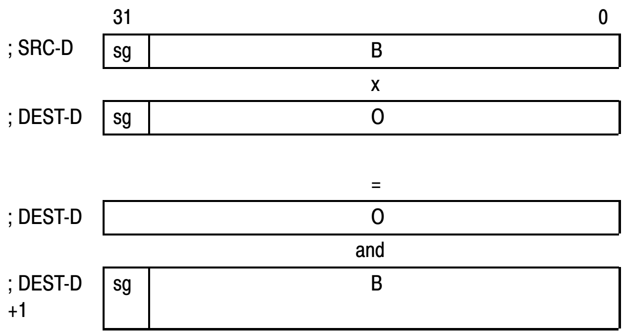
Byte, word, multiplication

MUL B/W B , O



Double word multiplication

MUL D B , O

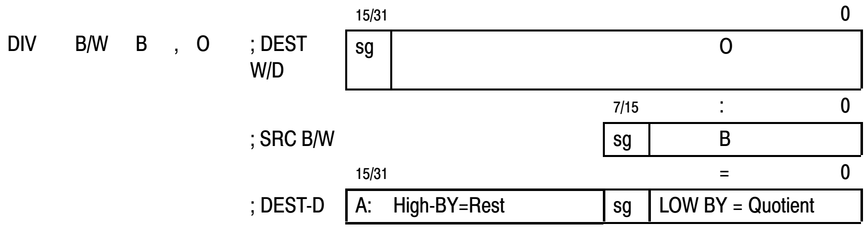


7.18.4 Divide instructions

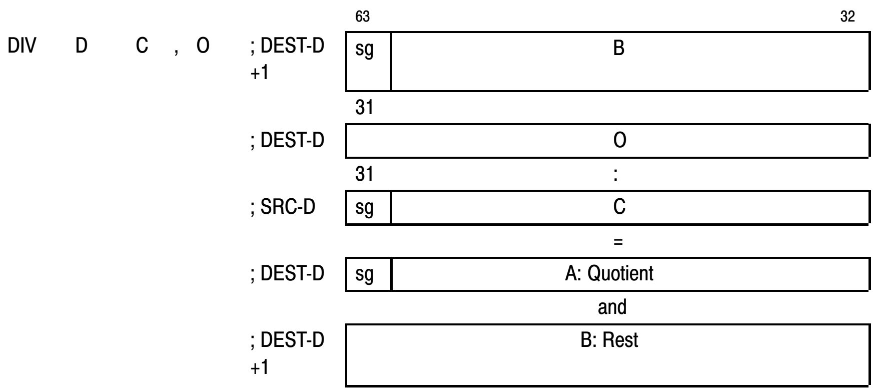
In division, the dividend always occupies the double width of the divisor.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
DIV	B W D	K R	, R			•	 •			0 0	• •	• •	• •	DIV B 100,A DIV W B,A DIV D B,A	Fixed-point division of signed integers.

Byte, word division



Double-word division



☞ **In the case of a division by 0, the division instruction is not carried out, and the overflow bit is set HIGH. The overflow bit is also set HIGH in the case of division overflow.**

7.19 Floating point arithmetic

Data formats, accuracy

Floating point arithmetic supports the data formats specified in the IEEE 754 and IEE 854 standards.
Two data formats, **REAL** and **LREAL**, are defined in accordance with IEC1131.

Data format	Data width	Mantissa	Exponent	Range
REAL: Short real floating point number, single precision	32 bits	24 bits	8 bits	$10^{\pm 38}$
LREAL: Long real floating point number, double precision	64 bits	53 bits	11 bits	$10^{\pm 308}$

Data format L always uses the register pairs AB and CD.

When calculating with the REAL data format, inaccuracies in the decimal range will occur sooner than with the LREAL format. If a high degree of accuracy is required, the LREAL format should be used.

☞ **Appropriate conversion routines are available in WinSPS from version 2.4 onwards.**

Value range and resolution

The floating point formats do not permit the representation of all numbers in any desired resolution. For example, if one wants to work with a unit of measure such as ⍰m, which is quite common in mechanical engineering, the REAL data format permits, for each individual μm, a representation with a limit value of 16.0 metres. If the LREAL format is chosen instead, the representation of numbers up to 17,179,869,184.0 m becomes possible.

Resolution		Value limit	
Floating point notation	Exponential notation	REAL	LREAL
1,0	E^{0}	16.777.228,0	18.014.398.509.481.984,0
0,1	E^{-1}	1.048.576,0	1.125.899.906.842.624,0
0,01	E^{-2}	131.072,0	140.737.488.355.328,0
0,001	E^{-3} milli (m)	16.384,0	17.592.186.044.416,0
0,0001	E^{-4}	1.024,0	1.099.511.627.776,0
0,00001	E^{-5}	128,0	137.438.953.472,0
0,000001	E^{-6} micro (μ)	16,0	17.179.869.184,0
0,0000001	E^{-7}	1,0	1.073.741.824,0
0,00000001	E^{-8}	0,125	134.217.728,0
0,000000001	E^{-9} nano (n)	0,015625	16.777.216,0
0,0000000001	E^{-10}	0,000976563	1.048.576,0
0,00000000001	E^{-11}		131.072,0
0,000000000001	E^{-12} pico (p)		16.384,0
0,0000000000001	E^{-13}		1.024,0
0,00000000000001	E^{-14}		128,0
0,000000000000001	E^{-15} femto (f)		16,0
0,0000000000000001	E^{-16}		1,0
0,00000000000000001	E^{-17}		0,125
0,000000000000000001	E^{-18} atto (a)		0,015625

Operands

Depending on the instruction, the following may be used as floating point operands:

- M, S, DM, DF, D, DXwith both direct and indirect addressing.

  The specified operand address must
  - be divisible as follows: by 4 for REAL data format and
  - by 8 for LREAL data format.
- K, register

☞ **P: A PM parameter may not be used as a floating-point constant. In the event that this is required, the constant may first be loaded into a marker word, for example.**

Instructions

The floating-point data formats and operands may be used in the following instruction types:

- LOAD floating point value
- TRANSFER floating point value
- CONVERT
- COMPARE floating point values
- Basic arithmetic functions
- Forming absolute value
- Extracting square root
- Logarithmic functions
- Trigonometric functions

Error displays, range overlaps

FPU errors (division by 0, ...) cause an “Error stop” in the iPCL führen in der iPCL, range overlaps cause a “Stop”.

7.19.1 Loadfloating point values

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
L	R	K	, R			•								L R 12,321,A	REAL constant → Reg. A
	L	R					•							L L A,C	LREAL reg. pair AB → CD
		M, S,				•								L L M8,C	LEAL M8-M15 → Reg. pair CD
		DP, DF, D, DX				•								L R DF16,B	REAL DF16-DF23 → Reg. B
		P				•								L R P0,D	REAL P0 → Reg. D
		OPD[R]						•						L L D[A],C	LREAL contents of operand addressed by reg. A → reg. C

7.19.2 TRANSFERfloating point values

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
T	R L	R	, M, S, , DP, DF, D, DX , P , OPD[R]			• • •		 •						T R A,M0 T L A,DF0 T R D,P0 T L D,[A]	REAL reg. A → M0–M3 LREAL reg. pair AB → DF0–DF7 REAL reg. D → P0 LREAL contents of reg. D to operand addressed by reg. A.

7.19.3 CONVERT number formats (floating point <-> integer)

- 32Converting 32-bit integer values to floating-point REAL / LREAL.
- Converting floating-point REAL / LREAL to 32-bit integer values.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
ITF	R L	R					•							ITF R O ITF L C	Converts 32-bit integer value from reg. A to REAL floating point format. Converts 32-bit integer value from reg. C to LREAL floating point format. The result is written to reg. pair CD.
FTI	R L	R					•							FTI R O FTI L C	Converts REAL floating point from reg. A to 32-bit integer value. Converts LREAL floating point from reg. pair CD to 32-bit integer value. The result is written to reg. C.

7.19.4 Convert data formats (REAL <--> LREAL)

In the REAL data format, inaccuracies may occur in the positions after the decimal point. If better accuracy is required, the LREAL data format must be used. To handle the required data format conversion, specific conversion instructions are provided.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
RTL		R					•							RTL O RTL C	Converts the REAL value of register A to an LREAL value. Destination register pair = AB. Converts the REAL value of register C to an LREAL value. Destination register pair = CD.
LTR	R	R					•							LTR L O LTR L C	Converts the LREAL value of register pair AB to a REAL value.Destination register = A. Converts LREAL value in register pair CD to a REAL value.Destination register = C.

7.19.5 Removing decimal positions

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
TRC	R L	R					•							TRC R O TRC L C	Writes the value in register A back to register A but without decimal positions. Writes the value in register pair CD back to CD but without the decimal positions.

7.19.6 Comparefloating point values

Controller instruction				RG		Addr.			Flag							Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z	AG	LG		
CPLA	R	M/K	, R			•				•		•	•	•	•	CPLA R M4,A	Compare REAL M4 to M7 with register A.
	L	S/DF/ D/DX/DP				•				•		•	•	•	•	CPLA L D200,C	Compare LREAL D200 to D208 with register CD.
		R					•			•		•	•	•	•	CPLA L A,C	Compare LREAL register pair AB with CD.
		P				•		•		•		•	•	•	•	CPLA R P62,A	Compare REAL P62 with register A.
		OPD[R]								•		•	•	•	•	CPLA L M[C],A	Compare LREAL contents of operand addressed by register C with register pair AB.

When comparing the REAL and LREAL data formats, the flags require arithmetical interpretation.

Examples:

Compare DEST (A) with SRC (B)		**CPLA B,A** **Jump instruction**
Equal	A = B	JPZ
Unequal	A ≠ B	JPN
Less than	A < B	JPM
Less than / equal	A ≤ B	JPMZ
Greater than	A > B	JPAG
Greater than / equal	A ≥ B	SPP

☞ **When using the CPLA instruction, the evaluation of the compare results must always be programmed immediately following the compare instruction itself. The user is advised to bear in mind that binary operations will cause a modification of the state bits. Therefore, a compare result can be used only in a link. Following this, another CPLA instruction must again be programmed.**

☞ **With various resolutions (decimal positions) the compare operation in the REAL data format returns correct results only up to specific limit values.**

Resolution / Value limit

Resolution	Value limit
0,001953125	256,0000
0,03125000	2048,000
0,2500000	32768,00
2,000000	262144,0
32,00000	2097152

Example:

```
L       R    2048.00000,A
CPLA    R    2048.00009,A
```

The difference is not found, and the numbers are recognized as being equal, Z = 1.

For large numbers at high resolution the LREAL data format must be used.

7.19.7 Calculating with floating point values

For working with floating-point values, the following basic arithmetic functions are available:

- Addition
- Subtraction
- Multiplication
- Division

The instructions handling the four basic arithmetic functions calculate the contents of the destination register or register pair with the contents of the source operand. The results are always written to the destination register or register pair.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
ADD	R	M/K	, R			•			•		•	•	•	ADD R M2,A	REAL M2 to M5 plus reg. A contents.
SUB	L	S/DF/D/DX/DP				•			•		•	•	•	SUB L D200,C	LREAL reg. CD minus D200 to D208
		R					•		•		•	•	•	ADD L A,C	LREAL reg. pair AB + CD
		P				•			•		•	•	•	SUB R P62,A	REAL reg. A minus P62
		OPD[R]						•	•		•	•	•	SUB L M[C],A	LREAL contents of reg. pair AB minus operand addressed by reg. C.
MUL	R	K	, R			•			•		•	•	•	MUL R 123.45,A	REAL 123.45 multiplied by contents of reg. A
DIV	L	R					•		•		•	•	•	DIV L A,C	LREAL reg. pair AB divided by reg. pair CD

7.19.8 Forming absolute value

Absolute values are always formed using a register or register pair. The result is then placed in the same register or register pair as a signed integer.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
ABS	R L	R					•		•		•	•	•	ABS R O ABS L C	Return absolute value of REAL contents of reg A. Return absolute value of LRLEAL contents of reg. pair CD.

7.19.9 Extracting square root

Square root extraction always uses a register or register pair. The result is then written to the same register or register pair.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SQRT	R L	R					•		•		•	•	•	SQRT R O SQRT L C	Extract square root of REAL contents of register A. Extract square root of LRLEAL contents of register pair CD.

7.19.10 Exponentiation

For exponentiation XY, the following procedure is used:

- In REAL format, registers A and C are used, with register A holding the base, and C the exponent. The result is written to register A.
- In LREAL format, register pairs AB and CD are used, with AB holding the base, and CD the exponent. The result is written to register pair AB.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
POW	R L	R	R				•		•		•	•	•	POW R A,C POW L A,C	Exponentiate the REAL contents of register A with the REAL contents of register C. The result is written to register A. Exponentiate the LREAL contents of register pair AB with the LREAL contents of CD. The result is written to register pair AB.

7.19.11 Logarithmic functions

The instructions for logarithmic functions calculate the contents of a register or register pair. The results are always written to the destination register or register pair.

Implemented are:

- Natural logarithms
- Base-10 logarithms
- Forming exponential functions from base-10 (common) logarithms

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
LN LOG EXP	R L	R					•		•		•	•	•	LN R O LOG L C EXP R C	Form natural logarithm from REAL contents of register A. Form common logarithm from LREAL contents of register pair CD. Form exponential value from common logarithm of REAL contents of register C.

7.19.12 Trigonometric functions floating point

The instructions for trigonometric functions calculate the contents of a register or register pair. The results are always written to the destination register or register pair.

Implemented are:

- Sine, with entry in radian measure
- Cosine, with entry in radian measure
- Tangent, with entry in radian measure
- Arc sine, main value
- Arc cosine, main value
- Arc tangent, main value

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
SIN COS TAN ASIN ACOS ATAN	R L	R					•		•		•	•	•	SIN R O COS L C TAN R C ASIN R O ACOS L C ATAN R C	Form sine from REAL contents of register A. Form cosine from LREAL contents of register pair CD. Form tangent from REAL contents of register C. Form arc sine from REAL contents of register A. Form arc cosine from LREAL contents of register pair CD. Form arc tangent from REAL contents of register C.

7.20 Parameter assignments

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
Pn	B W D	I/O/M/T/C/K S/SM/SYM D/DX/DF/DP FC/DM				• • • •								P0 W I0.0 P1 W S0 P2 D0 P3 PM0	Parameter definition for parameterized module calls. n: 0–62

7.21 Local symbol names & auxiliary markers for program tracking

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
DEF		I/O/M/T/C/K S/SM/SYM D/DX/DF/DP FC/DM	, SYM											DEF I0.0,-Symbol DEF I0,-Name	Definition of symbolic names that are locally valid only within the module in which they have been entered. Essential for the creation of library modules.
*		n n = 0–63												*1 1	Definition of auxiliary markers for program tracking. Processing of these auxiliary markers is written only to the marker buffer, and can be interpreted only in case of an error. *N has no influence on the program.

7.22 System variable

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
DEFW	W	K												DEFW W K0000H	Definition of function for system variable in OM2.

7.23 Jump instructions

Jump instructions may be executed unconditionally, and also in dependence on a binary link and/or a mathematical operation (see also Section 7.2 Flags). With one exception (JP [R]), jump instructions are programmed symbolically, but the entry point may not be located within a program branch because this would also cause the RES at the jump origin point to be linked.

Controller instruction				RG		Addr.			Flag							Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	C	O	N	Z	AG	LG		
JP		SYM [R],n*				•		•								JP -LABEL1 JP [A]	Unconditional to -LABEL destination. Unconditional by jump distance (byte) in register A.
JPB		SYM			•	•			1							JPB -LABEL2	Conditional, see flags.
JPCI		SYM			•	•			0							JPCI -LABEL3	Conditional, see flags.
JPCY		SYM				•				1						JPCY -LABEL4	Conditional, see flags.
JPCN		SYM				•				0						JPCN -LABEL5	Conditional, see flags.
JPO		SYM				•					1					JPO -LABEL6	Conditional, see flags.
JPON		SYM				•					0					JPON -LABEL7	Conditional, see flags.
JPM		SYM				•						1				JPM -LABEL8	Conditional, see flags.
SPP		SYM				•						0				SPP -LABEL9	Conditional, see flags.
JPZ		SYM				•							1			JPZ -LABEL10	Conditional, see flags.
JPN		SYM				•							0			JPN -LABEL11	Conditional, see flags.
JPAG		SYM				•								1		JPAG -LABEL12	Conditional, see flags.
JPMZ		SYM				•								0		JPMZ -LABEL13	Conditional, see flags.
JPLG		SYM				•									1	JPLG -LABEL14	Conditional, see flags.
JPCZ		SYM				•									0	JPCZ -LABEL15	Conditional, see flags.

The JP [R] instruction causes an unconditional jump whose jump destination must always be a jump instruction. This instruction variant was created specifically for the simple implementation of jump distributors. The controller monitors the instruction mnemonics of the entry point, and enters STOP mode if this fails to correspond to a jump instruction. In such cases, the error status of the Programming Unit (PG) provides information about the cause of the error.

The parameter n can be specified for the purpose of jump sequence monitoring, i.e. n can be less than or equal to the jump count.

The following example demonstrates the application of this jump instruction.

Example:

PLC-program interlude

Fixed program sequence

Jump distance calculation in register A for the following jump sequence A may only have odd-numbered values (1, 3, 5, ...). The parameter n must not be less than the following jump count.

```
JP      [A],n           ; 1 word instruction
JP      Dest1           ; 2 word instruction
JP      Dest2           ; 2 word instruction
   :
   :
JP      Destn           ; 2 word instruction
```

```
Dest1:                  ; Program part 1

PLC program

JP      End
```

```
Dest2:                  ; Program part 2

PLC program

JP      End
```

:
:
:
:

```
Destn:                  ; Program part n

PLC program

JP      End
```

```
:
:
End
PLC successor program
:
```

7.24 Module calls

Module call instructions may be executed unconditionally, and also in dependence on a binary link and/or a mathematical operation (see also Section 7.2 Flags).

The iPCL supports a module nesting depth of 63 program modules.

Two data modules may be kept enabled at the same time. For this purpose the following module calls are available:

CM, BAB, BAI DMn: enables DMn as 1st DM
BX, BXB, BAI DMy: enables DMy as 2nd DM

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
CM BX		DM PM PM P P PM[R]	, n , n			• • • • •		•						CM DM0 CM PM0 CM PM1,2 CM P0 CM P0,2 CM PM[A]	Unconditional, direct. Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMC BXB		DM PM PM P P PM[R]	, n , n		•	• • • • •		•	1					CMC DM0 CMC PM0 CMC PM1,2 CMC P0 CMC P0,2 CMC PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMCI BXI		DM PM PM P P PM[R]	, n , n		•	• • • • •		•	0					CMCI DM0 CMCI PM0 CMCI PM1,2 CMCI P0 CMCI P0,2 CMCI PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMCY		DM PM PM P P PM[R]	, n , n			• • • • •		•		1				CMCY DM0 CMCY PM0 CMCY PM1,2 CMCY P0 CMCY P0,2 CMCY PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMCN		DM PM PM P P PM[R]	, n , n			• • • • •		•		0				CMCN DM0 CMCN PM0 CMCN PM1,2 CMCN P0 CMCN P0,2 CMCN PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMO		DM PM PM P P PM[R]	, n , n			• • • • •		•			1			CMO DM0 CMO PM0 CMO PM1,2 CMO P0 CMO P0,2 CMO PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect
CMON		DM PM PM P P PM[R]	, n , n			• • • • •		•			0			CMON DM0 CMON PM0 CMON PM1,2 BAPN P0 CMON P0,2 CMON PM[A]	Conditional, see flags. Direct Parameterized, list to follow. As parameter. As parameter; para., list to fol- low. Indirect

Module calls continued

Controller instruction				RG		Addr.			Flag							Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z	AG	LG		
CMM		DM	, n			•										CMM DM0	Conditional, see flags.
		PM	, n			•										CMM PM0	Direct
		PM				•										CMM PM1,2	Parameterized, list to follow.
		P				•										CMM P0	As parameter.
		P				•										CMM P0,2	As parameter, parameterized.
		[R]						•								CMM PM[A]	Indirect
CMP		DM	, n			•			1							CMP DM0	Conditional, see flags.
		PM	, n			•										CMP PM0	Direct
		PM				•										CMP PM1,2	Parameterized, list to follow.
		P				•										CMP P0	As parameter.
		P				•										CMP P0,2	As parameter, parameterized.
		[R]						•								CMP PM[A]	Indirect
CMZ		DM	, n			•			0							CMZ DM0	Conditional, see flags.
		PM	, n			•										CMZ PM0	Direct
		PM				•										CMZ PM1,2	Parameterized, list to follow.
		P				•										CMZ P0	As parameter.
		P				•										CMZ P0,2	As parameter, parameterized.
		PM[R]						•								CMZ PM[A]	Indirect
CMN		DM	, n			•				1						CMN DM0	Conditional, see flags.
		PM	, n			•										CMN PM0	Direct
		PM				•										CMN PM1,2	Parameterized, list to follow.
		P				•										CMN P0	As parameter.
		P				•										CMN P0,2	As parameter, parameterized.
		PM[R]						•								CMN PM[A]	Indirect
CMAG		DM	, n			•				0				1		CMAG DM0	Conditional, see flags.
		PM	, n			•										CMAG PM0	Direct
		PM				•										CMAG PM1,2	Parameterized, list to follow.
		P				•										CMAG P0	As parameter.
		P				•										CMAG P0,2	As parameter, parameterized.
		PM[R]						•								CMAG PM[A]	Indirect
CMMZ		DM	, n			•					1			0		CMMZ DM0	Conditional, see flags.
		PM	, n			•										CMMZ PM0	Direct
		PM				•										CMMZ PM1,2	Parameterized, list to follow.
		P				•										CMMZ P0	As parameter.
		P				•										CMMZ P0,2	As parameter, parameterized.
		PM[R]						•								CMMZ PM[A]	Indirect
CMLG		DM	, n			•					0				1	CMLG DM0	Conditional, see flags.
		PM	, n			•										CMLG PM0	Direct
		PM				•										CMLG PM1,2	Parameterized, list to follow.
		P				•										CMLG P0	As parameter.
		P				•										CMLG P0,2	As parameter, parameterized.
		Fc[R]						•								CMLG PM[A]	Indirect
CMCZ		DM	, n			•					0				0	CMCZ DM0	Conditional, see flags.
		PM	, n			•										CMCZ PM0	Direct
		PM				•										CMCZ PM1,2	Parameterized, list to follow.
		P				•										CMCZ P0	As parameter.
		P				•										CMCZ P0,2	As parameter, parameterized.
		PM[R]						•								CMCZ PM[A]	Indirect

7.25 End of module instruction

End of module instructions may be executed unconditionally, and also in dependence on a binary link and/or a mathematical operation (see also Section 7.2 Flags).

Controller instruction				RG		Addr. type			Flag							Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	C	O	M	Z	AG	LG		
EM																EM	Unconditional
EMC									1							EMC	Conditional, see flags.
BEI									0							BEI	Conditional, see flags.
EMCY										1						EMCY	Conditional, see flags.
EMCN										0						EMCN	Conditional, see flags.
EMO											1					EMO	Conditional, see flags.
EMON											0					EMON	Conditional, see flags.
EEM												1				EEM	Conditional, see flags.
EMP												0				EMP	Conditional, see flags.
EMZ													1			EMZ	Conditional, see flags.
EMN													0			EMN	Conditional, see flags.
EMAG														1		EMAG	Conditional, see flags.
EMMZ														0		EMMZ	Conditional, see flags.
EMLG															1	EMLG	Conditional, see flags.
EMCZ															0	EMCZ	Conditional, see flags.

7.26 FIFO instructions

The iPCL provides four FIFO buffers, designated FI0 through FI3.

Each FIFO buffer has a size of 1024 bytes.

Reading from and writing to the FIFO buffers is accomplished with the LFI and TFI instructions. A single instruction reads or writes 1 to 32 bytes.

The number of bytes to be handled by the LFI / TFI instruction is variable, and is specified in Register C.

☞ **In the event that register contents are written to or read from FIFO buffers, the number of bytes will be defined via the operand attribute W/BY. Accordingly, operand attribute BY = one byte; operand attribute W = two bytes.**

When the number of bytes to be handled is variably specified in register C, each FIFO byte that is read or written causes the value in register C to be decremented.

In the case of a FIFO buffer overflow or underflow, the value stored in register C provides information about the number of bytes that could no longer be read or written.

FIFO overflow or underflow will not automatically cause a ZS STOP. As an indication of a FIFO overflow, carry bit SM31.3 is set HIGH. A FIFO underflow causes zero bit SM31.7 to be set HIGH.

The FIFO buffer is flushed with the RFI (Reset FIFO) instruction.

☞ **In the PNC all FIFOs are residual.**
In the osa master P-L/XL all FIFOs are not residual.

All FIFO instructions are RES-independent.

Controller instruction				RG		Addr.			Flag					Example			Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z				
LFI	B	FIn	, M/S/SYM			•	•	•		Ü:•			U:•	LFI	B	FI2,Df30	Read from FIFO buffer.
	W		, D/DX			•				Ü:•			U:•	LFI	B	FI3,D[A]	Number of bytes in C.
	D		, DF/DP			•				Ü:•			U:•	LFI	W	FI0,A	1 byte from FIFO into register A
	R		, OPD[R]							Ü:•			U:•	LFI	D	FI0,A	2 bytes from FIFO into register A
	L		, R							Ü:•			U:•	LFI		FI0,A	4 bytes from FIFO into register A
TFI	B	M/S/SYM	, FIn			•	•	•		Ü:•			U:•	TFI	B	DF0,FI2	Write to FIFO buffer.
	W	D/DX				•				Ü:•			U:•	TFI	B	D[A],FI3	Number of bytes in C.
	D	DF				•				Ü:•			U:•	TFI	B	A,FI0	1 byte from register A into FIFO.
	R	[R]								Ü:•			U:•	TFI	W	A,FI0	2 bytes from register A into FIFO.
	L	R								Ü:•			U:•	TFI	D	A,FI0	4 bytes from register A into FIFO.
RFI		FIn												RFI		FI0	Flush FIFO buffer.

7.27 Block commands

Block commands are provided as a convenient means of loading and transferring and also comparing and searching data blocks within the iPCL. The maximum size of these data blocks is 512 bytes / 256 words / 128 double words. The operand attribute indicates whether the block size refers to byte, word, double word, REAL or LREAL size.

☞ **The following minimum software version is a prerequisite for the use of the I and O operands in block commands: WinSPS 3.1**

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
BLT	B/W/D R/L	M/S I/O D/DX DP/DF OPD[B]	, M/S , O , D/DX , DP/DF , OPD[A]			• • • • •		 •						BLT B M0,D0 BLT W DF[B], M[A]	Block transfer from SRC address -> DEST address. Block size in register C.
CFxx CBxx	B/W/D	M/S I/O D/DX DP/DF OPD[B]	, R , O , D/DX , DP/DF , OPD[A]			• • • • •		 •					•	CFZ W MO,D0 CBN B M[B],D[C]	Forward/ backward compare operation within block.
SFxx SBxx	B/W/D	K R	, R , O , D/DX , DP/DF , OPD[A]			• • • • •		 •					•	SFZ W 50,M20 SBLG B B,M[A]	Forward/ backward search operation within block.

Block transfer

Block transfers are accomplished by shifting data blocks of defined size, whereby the data blocks may not overlap. Block transfers use only ascending addresses (incremental).

Example 1:

```
CM        DM10        ; 1st DM
BX        DM9         ; 2nd DM
L     D   50,C        ; Block size = 50
BLT   W   D20,DX40    ; Copy 50 words from DM9/D20 to DM10/D40.
```

Example 2:

```
L     D   50,A        ; DEST address offset
L     D   50,B        ; SRC address offset
L     D   50,C        ; Block size = 50
BLT   D   DF[B],M[A]  ; Copy 50 double words from DF50 to M50.
```

Block COMPARE

Compare two data blocks.

If the compare condition is met processing is stopped and the number of uncompared bytes / words written to register C. When using prefix addressing also the operand addresses are output to registers A and B.
The zero flag is set to HIGH if the compare conditions were not met throughout the entire range.

Block compare operations are possible in forward direction on ascending addresses, and in backward direction on descending addresses.

By interpreting the flags C, M and Z and their respective combinations, 8 compare criteria are available.

OPP	Description
	Compare forw. operation for the following:
CFZ	Equal
CFN	Unequal
CFAG	Arithmetical greater
CFM	Arithmetical less
CFLG	Logical greater
CFCY	Logical less
CFCN	Logical greater or equal
CFCZ	Logical less or equal
	Compare backw. operat. for the following:
CBZ	Equal
CBN	Unequal
CBAG	Arithmetical greater
CBM	Arithmetical less
CBLG	Logical greater
CBCY	Logical less
CBCN	Logical greater or equal
CBCZ	Logical less or equal

DEST block address direct or in register A, SRC block address direct or in register B, block size always in register C.

Example 1:

```
CM        DM10        ; 1st DM
L     D   50,C        ; Block size = 50
CFLG  W   D20,M20     ; Compare forward 50 words for logical greater
;                       starting at DM10/D20 with marker from M20.
```

Example 2:

```
L     D   50,A        ; DEST address offset
L     D   50,B        ; SRC address offset
L     D   50,C        ; Block size = 50
CBZ   D   DF[B],M[A]  ; Compare backward 50 double words for equal
;                       starting at DF50 with marker from M50.
```

Result evaluation of compare condition:

- Not met: Z-flag = 1
- Met: Z-flag = 0
  - In example 2 register A contains the operand address in the DEST block.
  - In example 2 register B contains the operand address in the SRC block.
  - Register C contains the count of data that was yet not compared.

Block search

The function searches for a character within a data block.

If the character is found, the number of bytes / words that were not searched is stored in register C. With the use of prefix addressing, register A will also contain the operand address.

If the character was **not** found (search condition not met) throughout the entire range, the zero flag is set to HIGH.

Through the interpretation of flags C, M, and Z, and their respective combinations, 8 search criteria are available.

OPP	Description
	Search forward for character:
SFZ	Equal
SFN	Unequal
SFAG	Arithmetical greater
SFM	Arithmetical less
SFLG	Logical greater
SFCY	Logical less
SFCN	Logical greater or equal
SFCZ	Logical less or equal
	Search backward for character:
SBZ	Equal
SBN	Unequal
SBAG	Arithmetical greater
SBM	Arithmetical less
SBLG	Logical greater
SBCY	Logical less
SBCN	Logical greater or equal
SBCZ	Logical less or equal

Block start address direct or in register A, search value as constant or in register B, block size always in register C.

Example 1:

```
L    D    50,C        ; Block size = 50
SFLG B    35,M20      ; Search forward 50 bytes starting at M20
;                       for the value 35.
```

Example 2:

```
CM        DM10
L    D    10,C        ; Block size = 10
L    D    50,B        ; Search value
L    D    20,A        ; DEST address offset
SRZ   D    B,M[A]      ; Search backward 10 bytes starting at M20 for
the value 50.
```

Result evaluation of search operation:

- Not met: Z-flag = 1
- Met: Z-flag and = 0
  - In example 2 register A contains the operand address of the searched operand range.
  - Register C contains the count of data that was yet not searched.

7.28 Interrupt instructions for time-controlled processing

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
TIM		R	, TI			•								TIM A,TI	Transfers interrupt mask. Writes interrupt mask for enabling / disabling interrupts. The mask was first loaded into a register.
LIM		TI	, R			•								LIM TI,B	Load interrupt mask, defined interrupt mask.
EAI		TI				•								EAI TI	Enable interrupt group.
DAI		TII				•								DAI TI	Disable interrupt group.
LAI		TII	, R			•								LAI TI	Load interrupt register (read statuses).
RI		R	, TI			•								RI A,TI	Reset interrupts based on a mask that was previously loaded.

7.29 Program stop and program end

Controller instruction				RG		Addr. type			Flag					Example	Comment
OPP	OPA	SRC	Z-OPD	A	E	D	R	[R]	V	CY	O	N	Z		
HLT														HLT	Halt command Controller enters STOP mode, program address is entered into error stack and outputs are cleared (deleted).
PE														PE	Program end. I/O state is initialized and the program cycle starts again at the beginning. At least one EP instruction must be present.

7.30 Backing up and loading residual areas

Residual areas are dealt with using the following functions:

- **Backing up** residual iPCL data occurs by:
  - the program-controlled writing to the static RAM of the osa P-L/XL or
  - the automatic backup on shutdown to the hard disk of the base device via an area in dynamic RAM in the PNC.

- **Loading** residual data into the iPCL
  - from the static RAM of the osa P-L/XL or
  - from the hard disk (previously backed up there during the automatic shutdown (PNC))

The residual areas to be backed up or loaded correspond only to those data modules

- that have been identified by a residual identifier in the symbol file.
- Operands as per residual limits set in OM2.

See ”Selection of residual data for cyclical backup” for iPCL page 3–12.

In the case of markers and the data field, specific areas of the defined residual area (Offset, Number) can be specified for the backup / loading procedures.

Controller instruction				RG		Addr.			Flag					Example	Comment
OPP	**OPA**	**SRC**	**Z-OPD**	A	E	D	R	[R]	V	CY	O	N	Z		
RS		DMn												RS DM1	Back up DM1 to static RAM or hard disk.
		M,T,Z,												RS M	Back up residual area, as defined in OM2.
		DF,DP												RS DF	
		M,DF	OFF, Anz											RS M10,50	Residual from M10 up, backup of 50 bytes.
RL		DMn												RL DM1	Load DM1 from static RAM or from hard disk.
		M,T,Z,												RL M	Load residual area as defined in OM2.
		DF,DP												RL DF	
		M,DF	OFF, Anz											RL M10,50	Residual from M10 up, loads 50 bytes.

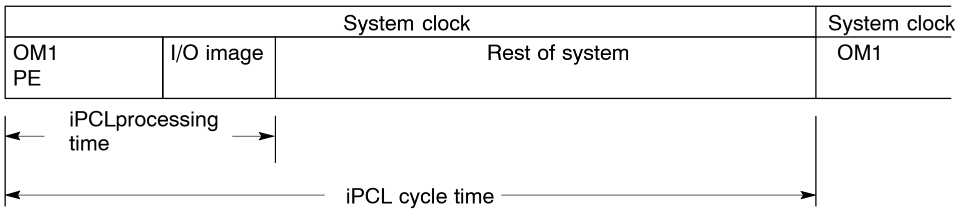
8 Processing Times

iPCL processing time

The iPCL processing time is the actual duration of program processing including the transfer of the I/O image to the bus master. Interruptions due to the interpolator and record change are included too.

iPCL cycle time

Die iPCL cycle time is defined as the time that elapses from the start of a program until the start of the next one.



Ratio of iPCL system clock to rest of system (Windows)

The PCL cycle time is started in a fixed time matrix that can be set in MACODA parameter 2060 00202.

In the MACODA parameter 2060 00211 the maximum proportion of processing time for the iPCL can be set. Up to how many percent of the total available computing time iPCL can take is set with this parameter. (Default value: 30%). If the PCL program exceeds this value, a warning message is generated indicating that the PCL program has left too little processing time for the rest of the system. Then either the time matrix of PCL starts must be increased or, if possible, the processing time of the PCL program must be decreased. iPCL continues to run during this.

Notes:

9 Sample Programs

9.1 Indirect addressing

```
                      ; DM checking whether
                      ; a) DM1-DM16 are present and
                      ; b) generating "existence bits" in result DM0/D0
                      ; c) Writing DM sizes into result DM starting with D2

L      D      1,A     ; Starting with DM1
L      D      0,B     ; DM existence bits in result DM in D0
L      D      2,C     ; DM sizes in result DM starting with D2
L      D      0,D     ; DM no. of result DM

CM     DM[D]          ; Result DM indirect module call

not_ready:
                      ; Check DMs and write results
U             DM[A]   ; Check DM, indirect module existence check
=             D[B]    ; If applicable, set existence bit HIGH (ind. bit
addressing)
PUSH   D      A       ; Save DM no.
L      D      DM[A]   ; Load DM size (indirect module size check)
T      W      A,D[C] ; Write to size word (indirect double word addressing)
POP    D      A       ; Write back DM no.
                      ; Increment address
INC    D      A,1     ; Next DM
INC    D      B,1     ; Next DM existence bit
INC    D      C,2     ; Next DM size word
                      ; All 16 DMs specified processed?
CPLA   D      16,A
JPCZ   not_ready ; jump on less than or equal
EM
```

9.2 Compare instruction examples

```
                                    ; Simuated compare value
                                    ;----------------------------
L      W    M0,A                    ; Load markers M0-M1
INC    W    A,1                     ; Increment register
T      W    A,M0                    ; Write value into markers M0-M1


                                    ; 1. Compare for "equal"
                                    ;-----------------------------

L      W    M0,A                    ; Current M0-M1 status
CPLA   W    10000,A                 ; Value 10,000 reached?

                                    ; Interpretation via links
U      Z                            ; Value 10,000 reached!
CU     Z0                           ; Increment counter Z0 by 1

                                    ; Interpretation via jump instruction
JPN    nicht_0
L      W    0,A                     ; Upon attaining the value 10,000
T      W    A,M0                    ; ... delete M0-M1
not_0:



                                    ; 2. Range monitoring
                                    ;------------------------

                                    ; Check value range 4000-6000
CPLA   W    4000,A
JPCY   Bereich_niO                  ; Value must not be less than 4000
CPLA   W    6000,A
JPCN   Bereich_niO                  ; ... and not greater than 6000

                                    ; Increment marker M2 in value window
4000-6000
L      D    M4,B                    ; Load markers M4-M7
INC    D    B,100                   ; Increment register
T      D    B,M4                    ; Write value to markers M4-M7

Area_nok:


                                    ; delete markers M0-M1 and counter C0 with
                                    ; trigger pulse
U      -RI_Anl
JPCI   no_RI
L      D    0,A                     ; Write value 0
T      W    A,M0                    ; ... to markers M0-M1
SC     A,Z0          ; ... and T0
T      D    A,M4                    ; ... on markers M4-M7
no_RI:
EM
```

9.3 FIFO instruction examples

```
DEF        SM31.1,-log1
DEF        SM31.6,-carry
DEF        SM31.7,-zero
DEF        M0.0,-trouble
DEF        M2.0,-nofifo
DEF        M6,-rest
DEF        M8.0,-RFI

                                  ; Tranferring data into a FIFO buffer:

BX         -db5                   ; Open data module

A          -nofifo                ; FIFO instruction locked?
JPB        end                    ; Then no transfer to FIFO

L      W   K30D,C                 ; 30 bytes from the second active
                                  ; DM
TFI    B   DX10,FI3               ; starting with D10 to be transferred to
FIFO FI3

A          -log1                  ; Lock FIFO instruction to prevent
                                  ; repeat execution
S          -nofifo

A          -carry                 ; FIFO overflow?
O          -zero                  ; FIFO underflow?
S          -trouble
JPCI       nosave
T      W   C,-rest                ; In case of an overflow / underflow the
count
                                  ; of the remaining data that could not be
nosave:                           ; transferred is written to
                                  ; register C.

L      W   C,C                    ; Monitor help
end:

                                  ; Delete FIFO:
A          -RFI                   ; Delete locked?
JPCI       noreset

RFI        FI3                    ; Delete FIFO FI3

A      B   -log1
R      B   -RFI                   ; Lock delete sequence to prevent
                                  ; repeat execution.
noreset:
```

Notes:

A Appendix

A.1 Abbreviations

Abbreviation	Description
BOF	Bosch Standard User Interface
DM	Data module
DRAM	Dynamic Random Access Memory
EM	End of module
EP	End of program
ESD	Electro-Static Discharge Abbreviation for all terms relating to electro-static discharge, e.g. ESD protection, ESD hazards, etc.
FBD	Function Block Diagram
IL	Instrction List, programming language
LD	Ladder Diagram, programming language
OM	Organisation module
PM	Program modules
RAM	Random Access Memory
SFC	Sequential Function Chart
SRAM	Static Random Access Memory
ST	Structured Text, programming language
TCP/IP	Transmission Control Protocol / Internet Protocol
UPS	Uninterruptible power supply

A.2 Index

Bosch Automation Technology

Australia
Robert Bosch (Australia) Pty. Ltd.
Head Office
Cnr. Centre - McNaughton Roads
P.O. Box 66
AUS-3168 Clayton, Victoria
Fax (03) 95 41 77 03

Great Britain
Robert Bosch Limited
Automation Technology Division
Meridian South
Meridian Business Park
GB-Braunstone Leicester LE3 2WY
Fax (01 16) 289 2878

Canada
Robert Bosch Corporation
Automation Technology Division
6811 Century Avenue
CAN-Mississauga, Ontario L5N 1R1
Fax (905) 5 42-42 81

USA
Robert Bosch Corporation
Automation Technology Division
Fluid Power Products
7505 Durand Avenue
USA-Racine, Wisconsin 53406
Fax (414) 5 54-81 03

Robert Bosch Corporation
Automation Technology Division
Factory Automation Products
816 East Third Street
USA-Buchanan, MI 49107
Fax (616) 6 95-53 63

Robert Bosch Corporation
Automation Technology Division
Industrial Electronic Products
40 Darling Drive
USA-Avon, CT 0 60 01-42 17
Fax (860) 4 09-70 80

Your concessionary

BOSCH

Robert Bosch GmbH
Geschäftsbereich
Automationstechnik
Antriebs- und Steuerungstechnik
Postfach 11 62
D-64701 Erbach
Fax +49 (0) 60 62 78-4 28

1070 073 875-101 (02.04) GB · HB NC · **BRC/ESM11** · Printed in Germany