PTP1000 System

User Manual

version 4.4E

tpa Tecnologie Prodotti per l'Automazione SpA

20099 Sesto S. Giovanni (Mi) Via Carducci, 221 tel.02/26223139 fax 02/2481008

To improve the quality and the performances of his products, TPA will be gratefull for any suggestion or warning about:

a) malfunctionning

b) documentation errors

c) any other errors

In the case of a) typology error, please specify the error environ and modes.

If some execution errors occurs, please supply the part program code, if possible.

In any case, specify the Software version, reading them from the Main Menu head.

All informations may be supplied by phone or fax.

1. INTRODUCTION

2. HARDWARE DESCRIPTION

3. THE OS1000 OPERATING SYSTEM

4. WORKING DISK

5. SYSTEM FUNCTIONALITY

6. THE GPL1000 LANGUAGE

7. MACHINE MANAGER

A. SYSTEM ERRORS

1. INTRODUCTION

GENERAL INFORMATION

The **PTP1000 system** is a **numerical control** including two main parts:

- a **Programming Unit**,for user interface, **Personal Computer** (**PC**) based.

- one or more (max 16, numbered from 0 to 15) **Central Unit Mainframes**, named **Modules**, including all the electronical devices for the machine controlling.

Every Module may include one or more electronical CPU cards (max 16), named **Stations**, with two possible typologies:

PTP200N	for two DC axes control
PLC200	for PLC functionality

The PTP200N card may be equipped (by a plug-in mounting) with an expansion auxiliary card, named **ESPAS**, for 3 other DC axes controlling and Analog Output driving.

Every stations may also include two type of **I/O Expansion** cards (**INOUTR** and **IOMOD** boards).

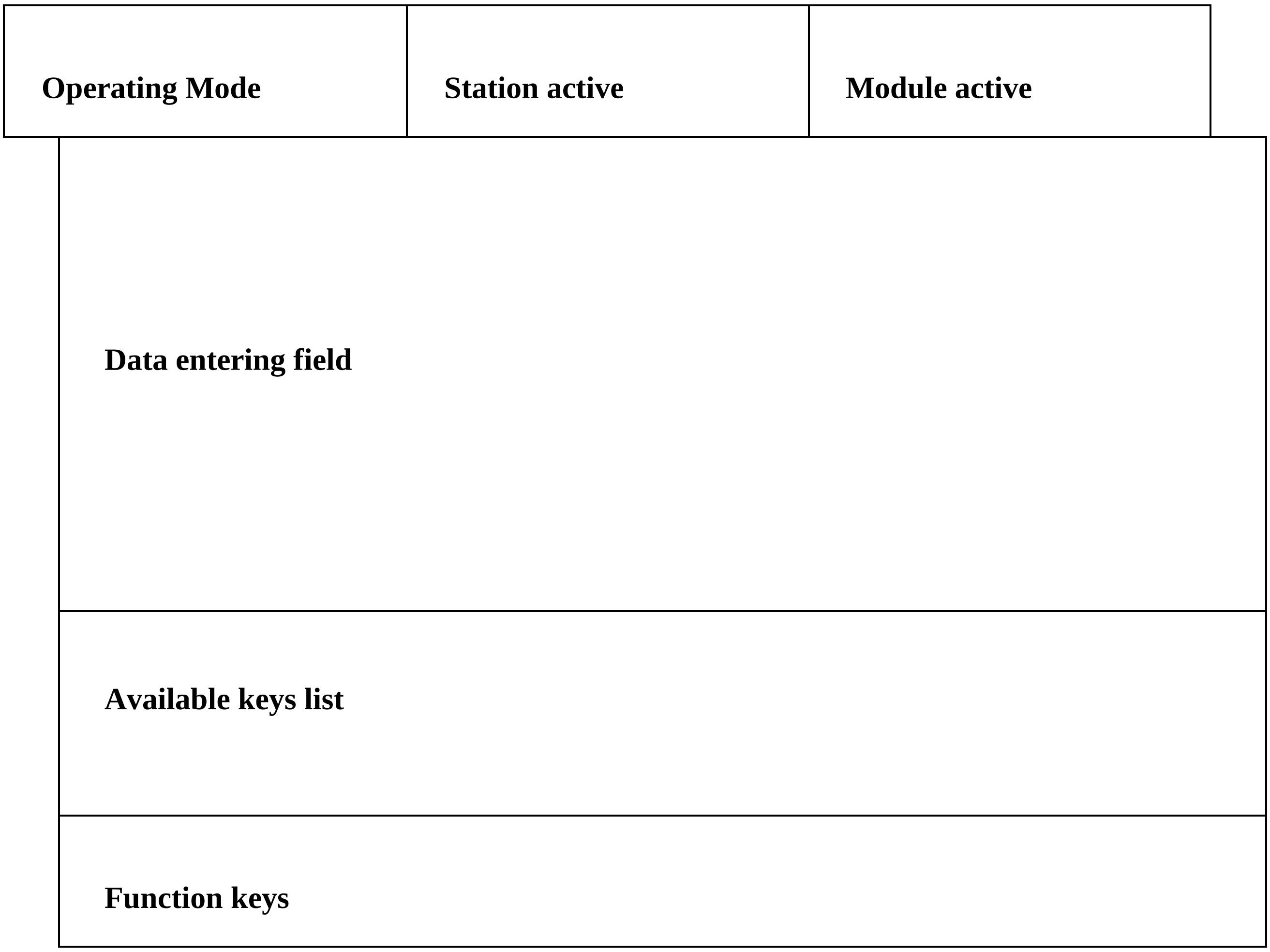
The comunication enter the PC unit and the Modules is achieved by a standard asynchronous **RS232 serial line** with a **Ring** connection (see also the RS232 Interface).

The Modules, on the PTP1000 numerical control, may used to drive one or more machines: then, by only one PC central unit it's possible to take under control many different working centers, independent each from others.

VIDEOSCREEN PAGE

The User interface is achieved, in the PTP1000 system, by the PC unit equipped by a special operating system, named OS1000.

The typical video screen page of OS1000 interface provides a different fields menu for the main operating modes, as showed in the following schematic picture:



MODULE AND STATION ACTIVE

In the different operating modes, on the first line on the screen, are displayed:

on the right the active Module description (in the all operating modes)

on the center the active Station description (only in some operating modes)

All the commands and data entering concerns the displayed Module and Station.

To select any other station or module, in any operating mode, the F9 - CHANGE softkey is available.

When a new Module is selected, its Station 0 is automaticaly enabled for displaying.

COUNTERS DESCRIPTION AND USING MODE

Every Station has at its disposal **thirtytwo 16 bit counters , numbered from 0 to 31**. These counters may be used by the programmer as events **counters**, **timers** or **index pointers** for the **Quotes Tables** using.

The counters map is the following:

Index number: General Purpose Counters or Timers

0	
11	

Index number: Tables of quotes or Counters or Timer

12	X axis table 0
13	X axis table 1
14	X axis table 2
15	X axis table 3
16	Y axis table 0
17	Y axis table 1
18	Y axis table 2
19	Y axis table 3
20	Z axis table 0
31	V axis table 3

If used as counter, every counter may assume a value from 0 to 65535.

If used as a timer, since the time base rate is 0.01 seconds, the maximum counting contents will be 655.35 sec.

INPUT / OUTPUT LINES NUMBERING

All the I/O lines are grouped in 8 bit (byte) **PORT.** Then any port includes 8 input or 8 output lines, addressed as the corresponding BIT number.

Then the **numbering** of the **bit (from 0 to 7)** defines the line position in the byte port. For this reason, in programming, any I/O line may be addressed by its corresponding **bit+port** code.

The **numbering** of the **ports (from 000 to 255)** may change according to the hardware setup and the **Maximum stations number** declared in the General Setup mode.

Related to the 2 differents available setups, about the Max number of station, the port organisation is the following:

- stations max number = 8:	16 Input ports (000÷015)
	16 Output ports (016÷031)
- stations max number = 16:	8 Input ports (000÷007)
	8 Output ports (008÷015)

In these two cases, the ports numbering refers to the station 0: for the others stations the ports numbering follows immediately, as showed in the next exemple, where two stations (0 and 1) are provided:

With a **station max number = 8** , we have:

station 0	16 input ports 000÷015
	16 output ports 016÷031
station 1	16 input ports 032÷047
	16 output ports 048÷063

With a **station max number =16** , we have:

station 0	8 input ports 000÷007
	8 output ports 008÷015
station 1	8 input ports 016÷023
	8 output ports 024÷031

If some mapped ports are not used by I/O Expansion cards, they may be used as virtual I/O lines for binary FLAGs in the programs.

Some Input/Output lines are compulsory reserved to the system for specific connections to Alarm signals, and precisely:

- in the Input: **ERRS** internal use
RDY internal use
SPEX refers the Setpoint procedure to define, in the
programs:
if = 1: Setpoint executed
if = 0: disables the Limit switches test (see
Emergency Table)
If SPEX is used as symbolic item, its name must
be included in the DEF file.
At the beginning of the Setpoint program, this
flag must be resetted by the RES SPEX instruction to
disable the axes limit switches test as emergency.
At the end of Setpoint, if no error occurs, this
line must be setted, by the SET SPEX instruction, to
restore the limit switches test.
At the beginning of any Automatic program, is
suggested to check the SPEX status to verify
if the Setpoint procedure has been correctly
executed.
MSGS internal use

- in the Output: **PWON** this signal is actived by the system at the end of
the Initializing procedure: this line is suggested to
be used to enable the machine Power On.
This signal is used only in the Station 0.
The phisical connection of the PWON signal
must be carry out on the OUT 23 of the Master card, and
precisely:
- num. max stations 8: bit 7 port 018
- num. max stations 16: bit 7 port 010
SYNC internal use.

CHARACTERS TO BE USED FOR DATA ENTERING

In the Data enter windows normaly some data typology suggestions or selectable fixed answers are supplied to user.

Other informations relate the unit of the measure or the numerical (intergers, decimals, and so on) or alphanumeric format.

For all these informations, the following conventional items have been assumed:

(/A/B) Fixed selection: the compulsory characters are indicated
withing the ( ) symbols, separated by the “/” character.

In this exemple, the selection must be made enter **space**, **A**
and **B** char. using the **Space Bar, Arrows Left/Rigth** keys.

(1_4) Selection enter a number sequence, with the min max value
separated by the **_** char**.**
In this case, a value from **1** to **4.**

(0.25_15/1,2,4,8) Two data are required (i.e. Feed Forward point to point); the
2 data are separed by the **/** char**.**
The first data must be entered choosing enter a decimal
number from 0.25 to 15.
The second data must be compulsory choosed enter the **1**, **2**,
4, **8** values.

(1_31*0.25_15) Two data are required (i.e. Feed Forward in interpolation);
the 2 data are separated by the ***** char.
The first data must be entered choosing an integer value from
1 to **31**.
The second data must be programmed as decimal value, from
0.25 to 15.

[pulses/mm] Unit of measure, enter the “[ ]” characters. In this case, the
unit of measure is:
encoder pulses per millimeter.

±x.x Data in decimal format, in this case with signed value
without particular limits.

x Data in integer format, unsigned.

bit+port or symbol Defines that the data may be entered in numeric or
mnemonic format:
numbers in the **bit+port** (i.e. 1005) case
symbol for mnemonic format (i.e. FC21).

AVAILABLE KEYS FOR DATA ENTERING

In the Data entry procedure, according to the data typology, only a group of keys are enabled, to prevent incorrect programming and syntax errors.

Then, for instance, for only integers entering, only the from 0 to 9 keys are enabled.

In other cases, the alphabetic characters, if typed in lower case, are automaticaly converted in upper case.

Other Function keys allows to achieve some specific functionality:

ENTER Ends the Data entering mode, confirming the programmed numerical values.

ESC Aborts the Data entering mode, cancelling all the programmed data.

BACK SPACE Cancels the character at the left of the cursor

DEL Cancels the pointed character.

INS Allows to insert new characters starting from the cursor position. It may be disabled retyping them.
The INS status is enhanced by the cursor in little square size.

Arrows Left/Right Move the cursor left / right.

Arrows Up/Down Move the cursor on the previous / next line.

HOME Move the cursor on the first character of the current line

END Move the cursor after the last character of the current line.

CTRL+HOME Cancels all the line modifying, restoring the previous data.

CTRL+DEL Erases all the characters in the line, moving the cursor at the beginning.

TAB Move the cursor to the next column, when the data enter window is organized on more columned fields.

SHIFT+TAB As in the previous case, move the cursor on the previous column.

MENU ITEM SELECTION

In the scrolling menu, to select the required item, the following keys are available:

Arrows Up/Down to move the cursor up/down, scrolling an item once.

PGUP/PGDN to scroll a multi page menu. In this case a vertical scrolling bar is displayed.

immediate search name; type, starting from the initial letters, the required item it’s possible to come back by the **BackSpace** key.

ENTER to selected the pointed item.

2. HARDWARE DESCRIPTION

PROGRAMMING UNIT

The programming unit is based on a **Personal Computer,** as Olivetti[1], IBM[2] or IBM compatible or on the Industrial PC mod. **COMPACT90** or **IMC90,** made by TPA; all based on the **MS-DOS**[3] operating system.

The minimum configuration of the PC is the following:

- 640K RAM
- Monochromatic Video monitor
- 40 Mb hard-disk and 1 1.44Mb Floppy-disk drive
- RS232 serial interface for asynchronous comunication with the PTP1000 central unit
- Parallel Centronics interface for printer connection

Optionaly, is suggested:

- 80Mb Hard disk or more
- Second RS232 serial port for Host-computer interfacing
- SVGA Color Monitor

CENTRAL UNIT

The Central Unit includes a modular 19" 6U rack mainframe (arranged for industrial cabinet mounting.) where may be inserted a set of boards (DIN requirements compatible), for machine interfacing.

The electronical boards set includes:

- the **PTP200N** CPU numerical control card;
- the **ESPAS** card, for DC axes expansion;
- the **HSINT** card, for high speed interpolation purpose;
- the **PLC200** CPU card, for PLC and AC axes driving functions
- the **INOUTR** card, for fixed I/O expansion
- the **IOMOD** card, for modular I/O expansion
- **power supply**

[1] . Olivetti is a trademark of Ing. C. Olivetti & C., SpA.

[2] . IBM is a registered trademark of the International Business Machines Corp.

[3] . MS-DOS is a registered trademark of the Microsoft Corp.

PTP200N

CPU card based on the 10MHz 8085AH1 microprocessor, including:

- 32K bactery including RAMCMOS
- 64K EEPROM
- 2 DC axes control devices
- 24 optocoupled digital Inputs
- 24 optocoupled transistor 24Vcc/1A digital Outputs
- RS232 serial port for PC interfacing
- parallel interface for high speed inter-CPU boards comunications
- plug-in connection for ESPAS card
- local motherboard interface for I/O expansion cards connection

ESPAS

Plug-in card compatible with the PTP200N CPU card, including:

- 3 DC axes control devices
- Auxiliary ±10V Analog Output
- 4 optocoupled input

HSINT

High speed interpolation board, based on 16 Mhz 80C186 microprocessor, including:

- 64K RAM
- 64K EEPROM
- plug-in connection for PTP200N interfacing.

PLC200

CPU card based on 10Mhz 8085AH1 microprocessor, including:

- 32K bactery including RAMCMOS
- 64K EEPROM
- RS232 serial port for PC interfacing
- parallel interface for high speed inter-CPU boards comunications
- 6 socket for modular I/O plug boards inserting (see IOMOD)
- local motherboard interface for I/O expansion cards connection

INOUTR

Fixed I/O expansion card, including:

- 24 optocoupled input
- 24 relais 24Vcc/2A, 220Vac/10A output

IOMOD

Modular I/O expansion card, including:

- 6 socket for I/O plug in boards inserting

The available modular plug-in boards, may be inserted on the PLC200 or IOMOD cards, are:

Modinp 8 optocoupled Input lines

Modoutr 8 Relais 24Vcc/2A, 110Vac/10A Output lines

Modainp 2 8 bit Analog/Digital converter lines

Modaout 2 8 bit Digital/Analog converter, with ±10V output range

Modcont[4] 3 16bit Encoder pulses Counters, for AC motors control

Modiofar Receiver/Trasmitter on Optical Fibre cable for Remote I/O devices interfacing.

The *remote modules* include:

Iofar	Remote Rx/Tx interface to Modiofar
Inp	8 Input module
Outpnp e **Outnpn**	8 PNP and NPN transistor 24Vcc/1A Output module
Outr	8 relais 24Vcc/2A,110Vac/10A Output module

[4] . The Modcont plug can be mounted only on PLC200 card.

POWER SUPPLY

Case version for 19” rack insercting.

- +5V	10A
- ±12V	2A

Stand alone, for direct cabinet mounting

- +5V	20A
- ±12V	2A

CE1004 MAINFRAME

Including 4 slots motherboard and power supply case.

dimensions: L. 332 H. 265 P. 275

CE1010 MAINFRAME

The same, but with 10 available slots.

dimensions: L. 483 H. 265 P. 275

This mainframe module, with external power supply box, provides 16 slots for cards inserting.

HARDWARE MODULES DESCRIPTION

The PTP1000 may be configurated in different modes, according to the required stations number. The availables architectures are the following:

Configuration with a maximum of 16 PTP200N or PLC200

This is the maximum module expansion for the 4 bit Parallel Interface capability.
The station CPU cards are adressed from 0 (MASTER card, in the left side of the mainframe) to 15.
In this case, the maximum I/O lines number of the module (1024 Input and 1024 Output) must be equaly in 64+64 I/O blocks per station. The I/O expansions may be achieved by an IOMOD or INOUTR cards, only one for station.

Configuration with a maximum of 8 PTP200N or PLC200

This is the standard and normaly suggested setup of the module.
The maximum I/O lines number is equaly divided in 128+128 I/O blocks per station.
In this case, a maximum 4 INOUTR or IOMOD expansion card may be connected to every CPU.

DC motors interfacing devices

On the PTP200 CPU card two DC motors interfacing circuits are provided, including:

- ±10V analog output, achieved by an 12 bit D/A converter, equipped with Offset and Gain regulation trimmers.

- 2 Optocoupled input lines for Encoder Phases signals, providing:
  - electronic *1, *2 or *4 hardware selectable multiplier
  - 50 Khz input max. frequency counting rate (200 Khz with *4 elect. multiplier)
  - 5 or 12 Vcc square wave in **push-pull** or **open collector encoder** signals compatibility
  - direct power supply from card to encoders, selectable, by jumpers, enter +5 or +12 V/100 mA.
- LSI hardware pulses counter and phase discriminator device (5 MHz sampling rate)

On the PTP200N card the acceleration ramps may be programmed (in milliseconds, from 0 to the max. speed) and a continous speed and position control loop is provided. More, to every CPU card, may be connected the 3 axes expansion card (ESPAS), to improuve the multiaxes controlling capability.

In the 16 stations configuration, the maximum number of controlling axes is then 80, in the 8 stations configuration, 40.

MAINFRAME CONFIGURATION

As described, the CE1010 mainframe includes the motherboard, for cards inserction, making available until 16 slots, or 10 if the power supply box is included.

In the motherboard, the following system buses are provided:

- Power supply voltages
- Parallel interface for inter-CPU comunication (PTP200N and PLC200), named (**ISB**)
- Local bus for CPU and I/O expansion cards (IOMOD or INOUTR) interfacing, named (**ILB**)
- Serial **RS232** interface

• Power supply

The power supply signals include the filtered +10V ±10%, ±20V ±10% voltages, multiplied on the motherboard (8+8) plug: then, in the case of external power supplier, its signals must be connected to the plug. This connection is no more needed if the power supplier is internal.

• Parallel Bus Interface (ISB)

Used for speedy data trasfer enter the CPU cards (PTP200 or PLC200), allowing a maximum of 16 cards comunication.

• Local Bus Interface (ILB)

Used for data trasfer enter every CPU card and the relative I/O expansion (INOUTR or IOMOD) cards.

• Serial RS232 Interface

Used for data trasfer enter the Mainframe CPU cards and the PC based Programming Unit.
This connection has a **RING** architecture, to provide multi point interfacing enter a variable number of Stations: the Tx signal from PC is connected to the Rx of the first left (Master) CPU card, the Tx of this last to the Rx of the second, and so on. The Tx of the last CPU is connected to the PC Rx line.

3. THE OS1000 OPERATING SYSTEM

OS1000 OPERATING SYSTEM DELIVERY

The PTP1000 operating system, named **OS1000**, is released on a 3.5” 1.44 Mb floppy-disk, compatible with the hard disk operating system MS-DOS version 3.3 and following.

The OS1000 floppy disk DO NOT include the MS-DOS system files: for this reason the following installing procedure must be observed by the user.

Disk contents

The OS1000 system disk is formatted into four main directories, including the following files:

1) Root directory

INSTALL.EXE	Installing and Setup programs

2) directory \PTP1000

MILLE.EXE	Main program
MENU.EXE	Main menu and Programs Manager operating modes
VIED.EXE	Editor and Compiler operating modes
CONFMAIN.EXE	Setup operating mode
PLANCIA.EXE	Console and Automatic op. modes
AUTOAP.EXE	Programming (Autolearning) op. mode
OSCI.EXE	Logic Analyzer op. mode
MESSAGES.EXE	Resident Messages manager
GPL1000.TPA	GPL1000 language macroinstructions Table
FONT3	Graphic character font

3) directory \MESSAGGI

Includes a different number of files, according to the installed national languages. The **TSIMB.lng** and **INFO.lng** files may be not presents.

MESSPTP.lng	messages file in the selected language (.lng). For instance, if lng=ITA(italian), will be MESSPTP.ITA.
MESSOSCI.lng	messages file for the Logical Analyzer operating mode.
ERRSYS.lng	System Errors messages file
TSIMB.lng	Function symbols Table file
INFO.lng	(optional) General information for installing file

LINGUE.MNU	Selectable language extension List file

4) directory \TOOLS

COMUNICA.EXE	Special Tool program for testing PC to module comunication
SPY.EXE	Serial RS232 line Monitoring program (requires an auxiliary external PC)
SPY.HLP	SPY using Help

The SPY.HLP file includes display or print DOS commands, as: TYPE and PRINT.

OS1000 INSTALLING and SETUP

The OS1000 installing procedure may be used for two purposes:

- installing the OS1000 system files on Hard Disk

- modifying the environ parameters (SETUP) included in the ENVIRON.TPA file.

During the first installing, the ENVIRON.TPA file (see later) is created and stored on the same disk.

Before beginnig the installing or setup procedure, it's suggested to consult the following User Manual sections:

- ENVIRON.TPA file structure (sect. 3)
- Working Disk Structure and Organisation (sect. 4)

Installing and Setup Procedure

The following procedure must be observed:

1) Set in **MS-DOS** at the prompt **A:>**.

2) Insert the **INSTALLING disk** in the drive **A:**.

3) Type: **A:INSTALL**

4) The **INSTALL** program is loaded in the disk and runned: if the LINGUE.MNU is present, the language menu is displayed on the screen to allows selecting of the required messages language.
   Then the box with the request **System on disk\directory** is presented.

   At this point, user must define the drive and the directory where the OS1000 operating system must be installed or where the setup data are stored (i.e.: C:\PTP1000): the directory name is free; system suggests C:\PTP1000.

5) The system reads the ENVIRON.TPA file on the selected disk: if missing, default data are assumed.

6) Then the **Working directoryes** are presented:

   - the System directory name,

   - the working directories names, for any module.

   It's possible to confirme or modify these names (i.e. if more machines working disks are stored in the disk) to access to the required working disk.

7) The **Languages Menu** is presented to select the default language, assumed in the system starting.

8) Then the **Setup Data** are presented, to confirme or modify, as:

   - Video screen typology

   - Start from AUTOEXEC.BAT

9) Then the Setup data are updated.

10) At the end, the **Execution of installing** request is presented:

    by typing NO the program stops without installing.

    by typing Yes, the installing procedure is started, copying the OS1000 system files in the destination drive.

Structure of the ENVIRON.TPA file

This file includes all the operating system installing and other utilities data for TPA environ. The environ data may be modified by the Setup procedure, excluding the data related to the colors used in the video pages: to modify them user must enter directly in the ENVIRON.TPA file editing, using any type of text editor.

In the file may be inserted comment lines, headed by the "**;**" character.

The ENVIRON.TPA file is an ASCII sequential file and includes:

• informations about the working disk

SYS = system directory name
0 = working directory name for the module 0
1 = " " " " " " 1
: :
: :
15 = " " " " " " 15

• informations about the user language

LINGUA = extension name of the MESSPTP message file: defines the language included in the files and then used by the system.
If, for instance, **LINGUA = ENG** the message file MESSPTP.ENG will be used (see the language list in the LINGUE.MNU file).

• informations for installing

AUTOEX = defines if the system is actived automaticaly by the PC at the power-on by the AUTOEXEC.BAT file (1) or not (0).
If **AUTOEX=1**, in the AUTOEXEC.BAT file will be present the command string : PTP1000.BAT.
If **AUTOEX=0**, the system will be runned by keyboard with the command PTP1000.

• informations about the OS1000 operating system

VIDEO = typology of the video screen (**0**=Monochromatic[BW]/**1**=Color[CO]).

COLOR = inform. number, background color followed by the foreground color, where:

inform. number is the number of information wich the colors refer.
color background is the background color code in hexadecimal.
color foreground is the foreground color code in hexadecimal.

• informations about the messages files

DIRLING = path name of the messages files, as:

- MESSPTP.lng
- MESSOSCI.lng

- MESS*.lng (auxiliary messages files)
- ERRSYS.lng
- GPL1000.TPA
- LINGUE.MNU

if DIRLING = no path, the current directory is assumed by default

The used **basic colors table** is the following:

N. info	**Cod.**	**Col. background**	**Col. foreground**	**Description**
0	07	BLACK	WHITE	main
1	0F	BLACK	BRIGHT WHITE	alternate
2	87	GREY	WHITE	keys area
3	6E	ORANGE	YELLOW	user errors msg
4	8A	GREY	BRIGHT GREEN	available keys
5	2F	GREEN	BRIGHT WHITE	VIDEO instr. msg
6	4E	RED	YELLOW	system errors msg
7	6E	ORANGE	YELLOW	cycle errors msg
8	13	BLUE	LIGHT BLUE	box frame
9	1E	BLUE	YELLOW	box title
10	1A	BLUE	BRIGHT GREEN	box main color
11	17	BLUE	WHITE	box altern. color
12	3F	LIGHT BLUE	BRIGHT WHITE	input main color
13	1F	BLUE	BRIGHT WHITE	input altern. color
14	3F	LIGHT BLUE	BRIGHT WHITE	reverse
15	1C	BLUE	BRIGHT RED	menu first character
16	30	LIGHT BLUE	BLACK	help
17	CF	BRIGHT RED	BRIGHT WHITE	row 1 of the screen
18	30	LIGHT BLUE	BLACK	function keys title
19	6E	ORANGE	YELLOW	pushed function key
20	1C	BLUE	BRIGHT RED	button off
21	2E	GREEN	YELLOW	button on
22	2E	WHITE	BLACK	non protected lines in Programming
23	2E	GREEN	BRIGHT WHITE	protected lines in Programming

If, for instance, user want to modify the main color (n. 0 in the table) choosing the BLACK color with the WHITE in background, in the ENVIRON.TPA file, must insert the line:

COLOR = 0,70.

The available **color codes** are:

BLACK	0
BLUE	1
GREEN	2
LIGHT BLUE	3
RED	4
MAGENTA	5
ORANGE	6
WHITE	7
GREY	8
BRIGHT BLUE	9
BRIGHT GREEN	A
BRIGHT LIGHT BLUE	B
BRIGHT RED	C
BRIGHT MAGENTA	D
YELLOW	E
BRIGHT WHITE	F

LINGUE.MNU file creating

This file includes all informations about the available languages: it is used when user access to the language Menu. This file may be edited with any text Editor program. The maximum number of supported languages is 10.

This file is released, with the OS1000 system disk, including the following languages:

Italiano,ITA
Español,ESP
English,ENG
Deutsch,DEU
Français,FRA

If the file is missing, the only available language is defined in the ENVIRON.TPA file.

Any record of the file includes two informations:

- **language name** is the name will be displayed in the Language selection menu (i.e.: English).

- **extension of the language** is the extension characterising the message files (i.e.: ENG).

The extension name is compulsory 3 character lenght.

If, for instance, a three language menu is required, the following three records must be inserted in the file:

Italiano,ITA
English,ENG
Deutsch,DEU

Consequently, thrre messages files must be provided:

MESSPTP.ITA
MESSPTP.ENG
MESSPTP.DEU

If a non existing message file language is attempted to select, the error message “Non available language” will be displayed.

If a custom LINGUE.MNU file is required in the OS1000 system file, released by TPA, user must copy them in the \MESSAGGI directory; in this mode, the custom file will be automaticaly installed by the initialising procedure.

4. WORKING DISK

WORKING DISK STRUCTURE AND ORGANISATION

The **Working disk** is the part of the PC hard disk where are stored all the user generated files.

The **Working disk** is divided in different areas (**Directories**): one of these, common for all the modules, is named **System directory**, the other areas, anyone reserved to a different module, are named **Module directory**.

The directory names may be modified by the user: the default standard names are:

\PTPSYS	for the system directory	
\PTP0	for the module directory	0
\PTP1	" " " " "	1
:	" " " " "	:
:	" " " " "	:
\PTP15	" " " " "	15

If a large capacity hard disk is provided by the user PC, many PTP1000 systems may be stored, into different directories selectable by the Setup procedure.

In the **System directory** the following files are stored:

CONFSYS.PAR	System Setup data.
OPTSYS.PAR	System Options data.
OPTMOD.PAR	Module Options data.
TAMP.DAT	Permanent memories status, used for re-transmissions.
FKEY.DAT	User defined Editor function keys.
BACKUP.DAT	Automatic Saving in Autolearning mode data (if used).

Every **Module directory** includes the following files:

CONFGEN.PAR	Module General Setup informations.
CONFSTAZ.PAR	Every stations Setup informations.
CONFCMP.PAR	Setup Compiled data, to be transmitted to the stations
DESCNT.PAR	Counters description data: these are present only if, in the
Setup, the	Counter Description option is selected.
CONFKEYB.PAR	CAT90 Autolearning console Setup parameters.
DIRSOR	User part-programs (source files) directory.

and the other files, as: User Part-programs, I/OR Definitions, Functions, and so on.

FILES NAMES STRUCTURE

Every PTP1000 generated file name includes a **Name** (max **8 alphanumeric** char.), followed by the **Station number** extension**.**

Exemple: SETP.0

where **SETP** is the program name for the **station 0**.

During saving, the system adds to the file name also the **typology** and converts the station number in *hexadecimal* code.

Then, if under MS-DOS a DIR command is recalled, the previous file name will be presented as: **SETP.0S** (where S defines the Setpoint typology).

If the name has been: **SETP.15** it should be traslated into: **SETP.FS**.

Some particular files require a fixed name to be recognized by the system, and precisely:

SETP	Setpoint	(typology: S)	
DEF	I/OR Definitions	(typology: D)	
ERRlng	Cycle errors	(typology: E)	where **lng** is the language extension
MSGlng	User messages	(typology: M)	where **lng** is the language extension

The other files names are free: more, the names of a group of programs or Tables of quotes, to be executed by different stations, may be equals or differents, for each station, according to the **Module Options** defined in the **Setup** file.

FILES TYPOLOGY

In the PTP1000 system, the user generated files may have different typology: at the storing on the disk, the names of the part programs are stored in an index file, named **Direttorio**, with the following auxiliary informations:

- **comment** to the file
- **date** of the last modifying
- the ***** mark to define, in the case of source programs or functions, if they have been **compiled**
- the **number of the bytes** that the program, in executable format, will keep busy in the card RAM memory.
- the **typology**

The files **typologies** are:

A	**Automatic** part program (working cycle)
S	**Setpoint** program
F	**Function** files
D	**I/OR Defnitions** files
Q	**Table of quotes**
G	**Drawing** file, generated by CAD system
E	**Cycle Errors**
R	**Report** files
T	**Generic Text** file
O	**Logic Analyzer** stored data files
M	Messages files to be recalled by the **Message** instructions

The typology codes are showed under the **T column** in the **Directory** listing.

The Typology organisation allows, in the different operating modes, the access to the required typology files, by the following menu:

Programs[1]
Setpoint[1-2]
Function[1]
I/OR Definitions[1-2]
Tables of quotes
Drawings
Cycle Errors[2]
Report
Text
Messages[1-2]
Analyzer

SET POINT AND AUTOMATIC PROGRAMS

User may generate two typologies of programs:

- Setpoint programs	(typology: S)
- Automatic programs	(typology: A)

The **Setpoint programs** have the **SETP** fixed name, one for every stations. They must be created only if, in the **Module Options** in the **General Setup**, the option **Setpoint operating use** is enabled, otherwise they will be ignored by the system. Having to fixed name, they are selected by the stations menu.

[1] . These files are compiled automatically before the source file saving; sorgente; then the compiled file (for Program, Setpoint and Function source files) include the executive code lines must be transmitted to the card during execution. Then for this tupology of files, besides the source file, also the compiled files will be stored in the Directory, marked by the (*) character in the C column.

[2] . These files typologies have a fixed name, then there selection is made by the corresponding station selectio.

The **Automatic programs** have an user-selectable name and may be more for every stations: user will select the required for execution. The Setpoint procedure may be included in the beginnng of the Automatic program: for this reason the Setpoint operating mode may be optional.

If in the **General setup file** the **Module option: Common names for all stations** is enabled, the name of the programs to be executed in contemporary, by every station, must be the same, as, for instance:

AUTO.0	for the station: 0
AUTO.1	for the station: 1

Every program usually includes:

- the PROG reserved word to identify the beginning of one of the parallel
programs (1÷8).

- GPL1000 instructions are the instructions to be used to make up a part
program (see also the Sect. 6 -GPL1000 Language)

- Functions call to recall a Function stored in another file (see also
Function File Description).

- comments free programmables string placed in the statement
line,using the "**;**" character as separator.

A part program , either of Setpoint or Automatic, may be subdivided in **8 programs** or **tasks**, to be executed in parallel (**multitasking**) mode, **numbered from 1 to 8**. The **program 1** is assumed as **main program** from wich the other used tasks will be actived. The 8 programs execution management and synchronisation is achieved by the **GPL1000 instructions.**

Every task must be headed by the **reserved declaration PROG statement** , followed by the identification number (**i.e. PROG** 2).

Exemple:

```
         PROG                  ;main program declare statem. (1 by default)
         APROG   2             ;program 2 exec. start
         |
         |
         APROG   2,START       ;prog. 2 started from Label
         |
         END

         PROG    2             ;program n. 2 declare statem.
         |
START:   |
         EPROG   2             ;program n. 2 end
```

As shown, it’s possible run a parallel program starting them from any labeled instruction: this means that, potentialy, everyone may be subdivided in many other short programs, to be actived according to the cycle requirements.

An alternative mode to active parallel process is to provide a main program recalling parallel tasks (max. 8) to be executed in contemporary. The main program will recall a parallel task by it name.

System provides 8 free tasks at disposition for the parallel programs requirements. In any moment, when the main program requires the execution of a lot of parallel tasks, this request will be accepted only if the available free tasks number is equal or greater of the required programs: on the contrary, system waits the end of other running tasks until the required number may be satisfacted. Then the program follows to the next instruction.

Every task begins with the reserved PROG item, followed by its name and ends, on the exit point, with the EPROG instruction.

Exemple:

```
         PROG                            ;main program (1) declaration statem
         |
CICLO:   X       30,Q
         Y 25, Z 18, Q
         APROG   Close, Decharge         ;two parallel prog. starting
         X       0,Q
         WPROG   Close                   ;parallel prog. end wait
         APROG   Reset                   ;parallel prog. starting
         Y 0, Z 0, Q
         BRA     CICLO
```

;parallel programs

```
        PROG      Close          ;parallel prog. heading declare
        |
        CloseCover               ;function
        |
        EPROG                    ;parallel program end

        PROG      Decharge       ;parallel prog. heading declare
        |
        |
        EPROG                    ;parallel program end

        PROG      Reset          ;parallel prog. heading declare
        |
        ResetRams                ;function
        |
        EPROG                    ;parallel program end
```

SYMBOLIC DEFINITIONS FILE

All the input and output lines, the flags and the other data used in PTP1000 programming may be defined in direct numerical format or as mnemomonic symbols, for better readability. All these symbols must be defined using special declare statements, and these definitions must be shared with all the executive programs and the Compiler.

The symbolic format may be used only for interger numbers exprimible data.

The symbolic form is not compulsory but, in particular for I/O signals, may be very helpful in programming and in all the Diagnostic procedures.

All the simbolic definitions must be stored in a particular file (one per station), having a fixed name (**DEF**) and **typology** = **D**. Having a fixed name, it is automaticaly selected by the station menu at the item **I/OR Definitions.**

The mnemonic name, for each symbol, has a free **lenght,** but user must make account that, in the Manual and Diagnostic operating mode, only the first 8 characters will be displayed: the symbolic definition **maximum number** is **500.**

In the file, any **text row** includes:

symbol **code** **;comment**

where:

- symbol is the mnemonic item associated to the data: may be used only alphanumeric or the “÷ **-** < > “ characters**.**

- code is the operation code, defining the symbol typology enter:

BYTE to identify a value from **0** to **255** or a **bit** (if the value is <= 7), for a **data**, a **port** or a general **value** in these formats:

decimal (i.e.: 7 15 121) Max: 255

hexadecimal if headed by the **$** symbol and followed by **H**
(i.e.: $1H $EAH) Max: $FFH

binary if headed by the **$** symbol and followed by **B**
(i.e.: $10B $01110011B) Max: $11111111B

any other BYTE type definition

BITPORT defines an input/output line or a flag, addressed by a value from **0** to **7** and by another from **0 to 255**, as:

4 decimal numbers the first defining the **bit 0÷7**, the others the **port 0÷255**
(i.e.: 2150 0001 7255)

a BYTE type value not greater than to 7, the “+” character and another **type BYTE value**
(i.e.: 5+122 3+EV BIT+$FFH)

any other BITPORT type value

DATAPORT defines the data + port address composition (see also the OUT Gpl instr.), as:

6 decimals numbers from **0÷255**, the first three digits, and the other three..
(i.e.: 012021)

to BYTE type value, the “+” char. and another BYTE type value

(i.e.: 57+98 $49H+$11B

DATA+PORT)

another DATAPORT type definition

STRING defines a general string message, closed enter the “ characters.

I.e. if the following string definition is provided:

SEPT_OK "Setpoint correctly ended"

in the GPL program the VIDEO instruction will may be writed as:

VIDEO SETP_OK

QUOTE defines a quote, compulsory with a decimal digit:

If, for instance, the following definition has been inserted:

STANDBY 10.0

in the GPL program, the corresponding moving instruction, may be writed as:

X STANDBY, Y STANDBY, Q

- comment is an optional string field, separed by the “**;**” character.

Then, in the Symbolic Definition file, we can have, for instance:

```
EV1     1010   ;bitport
EV2     2      ;bit
PRG2    2      ;value
NRIP1   50     ;value
ERR1    1      ;value
PORT10  10     ;port
MASK    125    ;data
ZERO    0      ;data
```

These symbols could be used, in the program, in the following mode:

```
PROG

APROG     PRG2
OUT       ZERO+PORT10,MASK
SET       EV1
REPEAT    NRIP1
|
|
SET       EV2+PORT10
|
|
ENDREP
|
|
ERROR     ERR1
```

FUNCTIONS FILE

General description

The functions (**typology: F**) must be considered as GPL subroutines recallable from any type of GPL main program (Setpoint or Automatic). Unlike the program internal subroutines, the Functions are permanently stored in the CPU cards memory, where have been unloaded during the **Initializing procedure** of the system or, if modified, during the re-transmission procedure.

Every function is identified by a **number** from **0** to **255**; then up to **256 functions** are available for every station. A function may be recalled many times by the same program, by the FCALL instruction, followed by the identification number or the mnemonic name (15 char. max lenght).

The functions may be store in one or more files, according to its specifical operating meaning.

To access, in **edit,** to the **functions** file, user must select, in the Edit menu, the **typology: Function** item.

The Function text must be headed by the **FUN** definition item, followed by the **number of the function (0÷255)**, by the **symbolic name,** if used, and, eventualy, by the **comment**.

Exemple:

```
FUN       10,OpenPliers ;comment to the function
|
|
        FRET
```

This function may be recalled, from the main program, as:

by the **number**: **FCALL 10**

by the **symbol**: **OpenPliers**

The return from function is achieved by the **FRET** instruction, inserted at the end of the function text.

It’s also possible call one or more functions as parallel program, using the FPROG instruction (see also the Multiprogramming description).

The following rules must be observed in the functions writing:

- to maximum of 4 functions nesting level is accepted and a function cannot recall itself.
- the APROG (active to program) instruction cannot be inserted in a function text.
- the function object code has a maximum lenght of 4095 bytes.
- a function cannot be present more times in the functions file or files of the same station.

Every function is compiled alone: then the programmer can repeat, in different functions, the same names for labels, without problems (as shown in the FUN 10 and FUN 20 of the following exemple).

During the functions transmission procedure, only the functions included in *Compiled* files will be sended to the stations cards.

Exemple of a functions files:

```
; ***  function 10  ***

            FUN      10,OpenPliers          ;opens the pliers

            RES      EV01
            SET      EV02
            WIZ      FC02,ERRFC02
            FRET

ERRFCO2:    VIDEO    LIMIT SWITCH FC02 NON DETECTED
            RES      EV02
            FRET
```

```
; ***  function 20  ***

            FUN      20

            SET      EV02
            WIZ      FC02,ERRFC02
            Y        6
            RES      EV02
            FRET

ERRFC02:    VIDEO    LIMIT SWITCH FC02 NON DETECTED
            RES      EV02
            FRET
```

Parametric Functions

The parametric functions, unlike the normals, allows to substitute to lot of the GPL1000 instructions arguments with a symbol with undefined value, every of which will be defined with the effective value only by the calling instruction.

Then every parameter, leaved undefined, will be marked to be presetted by the current value passed by the FCALL instruction, that must provide real values according to the position sequence of the arguments in the function text.

In the function text, the undefined arguments must be indicated by a mnemonic symbol headed by the **&** character: all these simbols must be listed in the **TSIMB.lng** file (see later); this file must be stored in the OS1000 directory.

If a symbol is used many times in the same function text, it will assume the same value at the calling moment, then, in the FCALL parameters list, must be defined only once.

Exemple:

We suppose to have the following parametric function:

```
FUN         05,Fparam          ;parametric function
SET         &OUTPUT
X           &QX
Y           &QY
WEND        XY,Q
RES         &OUTPUT
FRET
```

In the main program, this function may be recalled many times, anytimes passing to them different parameters values:

```
PROG

Fparam      EV1,12,25
Z           10,Q
Fparam      EV5,120,40.5
END
```

As shown, the OUTPUT symbol is passed once for every calling:then the Compiler will provide to expand in every statement using that parameter. The symbols maximum number is 60 per function.

During the function compilation, the system checks that all the used symbols are present in the TSIMB.lng file, verifying also the syntactic correctness and that all symbols have a typology compatible with the instruction argument, except for the interpolation instructions. For any argument, the Compiler reserves a number of bytes, initialized to zero, corresponding to the parameter typology (i.e.: for bit+port are reserved 2 bytes equal to 0); the total memory area for these parameters must be less then 256 bytes. For every function a description table, including the pointer and the lenght data, for every parameter, is created.

Function Directory

During the compilation of the functions, an index file, named **SYMFUN.nstationR**, (typology: R) is updated storing the list of the compiled functions.

This file is updated also everytimes a function is created, erased or modified.

In this directory file are saved, in sequence:

1) number (in decimal) of the function
2) function name
3) comment to the function (only the first 40 characters)
4) name of the file including the function
5) total number (in bytes) of the parameters area
6) symbols used in the function

alls separed by the *comma* character.

In the case of the previous exemple, we will have:

```
05,Fparam,parametric function,FUNCTION,12,OUTPUT,QX,QY,OUTPUT
```

In the Editor and Manual & Diagnostic operation mode, it’s possible to read the functions directory by the **F7 - FUN** softkey or to print them by the PRINT command in the Programs Manager oper. mode.

In the Editor are displayed the 1) to 4) informations.

In Manual & Diagnostic only the 1) to 3) info are displayed.

Table of the symbols may be used in the parametric functions

The parametric functions used symbols must be stored in the **TSIMB.lng** file, where .lng is related[3] to the current language: for english version will be: TSIMB.ENG.

An original version of the TSIMB.ITA file is released by TPA; if the user manages to custom TSIMB file, is suggested to make to backup copy of then, because during installation the original release, included in the OS1000 TPA system disk, will be alwais copied.

The TSIMB file must be alwais saved in the same directory of the OS1000 operating system: only one file is used by the system to stored these information, and may be modified by anyone text Editor. The first row must include the total number of used symbols: any other row include all the informations related to any symbol, and precisely:

1) symbol name with to 10 max. alphanum. characters, writed *without* the **&** char. used in the functions statement.

2) message is the message will be displayed during the data acquisition of the symbol, with 35 char. max lenght.

3) help if present, will be displayed together with the previous message for helping (40 char. max.).

4) argument number is an identification number for argument typology coding, as shown in the following list.

Alwais with reference to the previous exemple, in the TSIMB.lng file must be stored the following data:

```
3
OUTPUT,,,3
QX,,,50
QY,,,51
```

Arguments typology of the GPL1000 instructions

In the following table are listed all the argument typology must be used in the GPL1000 instructions: user must know these codes to compose correctly the TSIMB file.

Code	Value	Argument description
1	0÷7	bit
2	0÷255	data or port
3	0÷7+000÷255	bit+port
4	000÷255+000÷255	data+port

[3] .See also the language list in the “ENVIRON.TPA file structure in section 3.

5	0÷31	counter number
6	0÷65535	counter value
7	1÷65535	repeat number
8	1÷255	cycle error number
9	0÷31	offset number
10	0/1	no/nc
11	1÷8	tool number
12	0÷3	number of table of quotes
13	A/0÷3	number of table of quotes (TO=All)
14	</=/>	compare operator
15	A/R/D/F/Q	axes status
16	A/R/D/F/Q/+/-	axes status and direction
17	-10.000÷10.000	analog output voltage
18	x.x	time for Delay instruction
19	x	interpolation acceleration [in ms]
20	x.x	interpolation speed
21	XYZWV	axes names mask
22	XYZWV/I	axes names or interpolation mask
50	±x.x	quote of the axis X or X1
51	±x.x	quote of the axis Y or Y1
52	±x.x	quote of the axis Z or Z1
53	±x.x	quote of the axis W or W1
54	±x.x	quote of the axis V or V1
55	±x.x	quote of the axis U1
56	±x.x	quote of the axis X2
57	±x.x	quote of the axis Y2
58	±x.x	quote of the axis Z2
59	±x.x	quote of the axis W2
60	±x.x	quote of the axis V2
61	±x.x	quote of the axis U2
62	±x.x	quote of the axis X3
63	±x.x	quote of the axis Y3
64	±x.x	quote of the axis Z3
65	±x.x	quote of the axis W3
66	±x.x	quote of the axis V3
67	±x.x	quote of the axis U3
68	x.x	speed of the axis X
69	x.x	speed of the axis Y
70	x.x	speed of the axis Z
71	x.x	speed of the axis W
72	x.x	speed of the axis V
74	x	acceleration of the axis X
75	x	acceleration of the axis Y
76	x	acceleration of the axis Z
77	x	acceleration of the axis W
78	x	acceleration of the axis V

200	XYZWV	2D interpolating axes mask
201	XYZWV	3D interpolating axes mask
202	O/A	interpolation sense
203	x.x	interpolation speed
204	x.x	number of revol. in circ. interpolation
205	±x.x	quote or radius or pitch of interpolation

TABLES OF QUOTES

Preliminary

The Tables of quotes may be used only for DC axes moving.

Using the Tables of quotes (**typology: Q**) presents a lot of programming facilities, as:

- allows system using working only on the quotes layout, also without knowing the programming language.
- as in the case of repetitive displacement cycles, allows to implement many tables of quotes, with differents values (according to the working layout) making enchanged the program cycle (i.e. in the case of pallett load/unload robotic application).

Any Table of quotes file may include a **maximum** of **4 tables**, numbered from **0** to **3**, for any axis.

The quotes listed in the Table will be managed by apposite GPL1000 instructions.

It’s possible to store many Tables of quotes files, for every station, and select one of them to be linked to the Automatic program to be runned.

It’s also possible to link, to every step of the table execution and to a specific axis, one or more phisical output lines or flags, to be set or reset, according to the cycle requirements. The output line or flag will be drived before the corresponding axis movement.

Edit

Once the Table file is selected for editing, the following function keys are available:

F1 - DIM (dimensions)

Allows to define the table lenght, as number of table locations required by the application cycle.

The **total quotes number** is normaly limited by the following factors:

- the memory area assigned to the tables in the Setup,

- the number of bytes used to quotes coding (see also: **F4 - Option**).

F2 - OFFSET (zero offset)

Allows to defines to zero offset quote for *any axis* and for *any table*.
This offset is added to the current table location quote, when the EQTAB instruction is called for moving (if an absolute displacemnt is programmed).
In this mode, if required, it's possible to define the quotes layout from an initial point that may be different from the absolute Setpoint reference.

F3 - QUOTES (tables selection)

Allows to **insert** or **modify** the **quotes** and the corresponding **Output command**, if present, for one of the four tables.
If more tables are used, the table to be modified must be selected before.
If the table to be edited is new, all the quotes are set, by default, to zero.
In the case the table lenght is increased, by a following (F1 - DIM) command, only the new locations are initialized to zero; on the contrary, if the lenght is reduced, the exceeding locations are erased.
During the quotes editing, by the **F1 - EXPAND** function key, it's possible to insert, automaticaly, to lot of locations having, one from the other, the same distance (in mm.). If the **Out** field is pointed, this command replays the same functionality on the new locations.

F4 - OPTION (options)

Enables the following options for tables mangement:

number of bytes per quote Defines the quote lenght (in bytes) for any table location, according to the maximum displacement requirement for any step. For any table the selecteble values are: **1**, **2**, **4**.

Then, the corresponding **range**, in encoder **pulses**, will be:

1 byte = from -128 to 127
2 byte = from -32768 to 32767
4 byte = from -2147483648 to 2147483647

Generally, if no particular memory occupation problems are present, the **4 bytes value** is suggested (range assumed by default during a new table entry), to assure the maximum displacement **range.**

rate Used during a synchronized axes movements, defines the time slice for the incremental quote execution. This parameter (the 1 to 128 values, in quadratic progression, are admitted) is used in the following formula:

time slice = rate * 1 / real time frequency

where the real time frequency is normaly 200 Hz, then:

rate = 1	time slice = 5 ms
rate = 2	time slice = 10 ms
rate = 4	time slice = 20 ms
rate = 8	time slice = 40 ms ....
rate = 128	time slice = 640 ms.

Obviously, under the same incremental displacement, the highter will be the axis speed the less rate value will be programmed.

F5 - OUTPUT (Output parameter)

Allows to configure, for any table (0÷3), the Output type and the axis linking: the output will be set *before* the corresponding axis moving.
The parameters must be programmed, are:

Axis The following axis names are admitted:

(/X/Y/Z/W/V)

If the correspondence axis is missing, the output drive is ignored and in the F3 - QUOTES function key the Out column will be erased.
If the axis name is programmed, the output line will be actived *before* any axis displacement.

Output The following values must be entered:

(bit+port, mask+port, symbol)

where:

- bit+port defines the output line to be set or reset: in the column Out of the F3 - QUOTES user must select its quiescent status ON or OFF (assumed by default).

- mask+port is the bits mask of the Out port to be drived: in the Out column of the F3 - QUOTES function key, a value from 0 to 255 for the binary code must be entered. The default value is 0.

- symbol one of the previous two case but represented by a symbolic definition in the I/OR table.

F10 - EXIT (save and exit)

Stores the file, if modifyed, returning then to the Editor menu: Exiting by the ESC (escape) key, the new data are lost.

CYCLE ERRORS FILE

The Cycle Errors file (**typology: E**) stores all the errors messages included, by the user, in the Setpoint or Automatic programs or in the Functions: every message has a maximum lenght of 60 characters.

The file name must includes a fixed root **ERR** followed by 3 char. extension coding the corresponding language file: for every station, a specific file may be created.

For editing, user must select, in the stations menu, the **Cycle Error** file typology.

Editor loads the file corresponding to the current language. If a translation, from the italian file (ERRITA.0E) to, for instance, the english version is required, user must load the italian messages list, modify all the messages text, then select F10-exit and, last, by the command Save or Exit, when the file name is required, enters new name ERRENG.

In the part programs, the messages displaying is enabled by **ERROR** instruction where the numerical argument defines the message index.

When, during execution, some of these errors occurs, the corresponding task is suspended (if the ,C option is not programmed). The corresponding CPU card sends to PC the error number, allowing them to unload the error text from the station 0 file (using the number as pointer of the list) and to display the corresponding message: if the mnemonic error text is missing, only the error number will be displayed.

REPORT FILE

These files, with fixed names and **typology= R**, store a diagnostic informations directly produced by the system.

Any file are specific for every system Module and are selectables only by the relative Directory.

System provides 2 report files:

LSTCMP this file includes the object code of the last program compilation where the listfile on disk option has been selected.

REPORT this file includes a lot of errors/messages informations have been selected in the **Diagnostic Options** (in the **Module General Setup** operating mode) as:

- System or Cycle errors list
- VIDEO instructions produced messages
- Selected Counters contents.

TEXT FILE

The Text files (**typology: T**) are general purpose custom text files.

It’s possible create one or more files for any module, with free names.

MESSAGES FILES

The Messages files (**typology: M**) includes all the custom messages, recallable by the **MESSAGE** instructions, used in the Setpoint and Automatic program and in the Functions: any message has a maximum lenght of 32 characters.

The file name include a fixed root (**MSG**) followed by a 3 char. extension corresponding to the used language (i.e. MSGITA, MSGENG,..). For any Module, a maximum of 255 messages are admitted.

For editing operations, select, in the menu, the **Messages** file typology.

For translation requirements, user must follow the Cycle Error messages procedure, already described.

In the part programs, these stored messages may be recalled for displaying by the **MESSAGE** instruction, where the numeric arguments define:

- the *first*, the message table pointer

- the *second*, if present, the line number (1÷8) of the screen where the message must be displayed (if missing, the message will be displayed on the first line).[4]

[4] . For more informations about displaying, see also the Automatic operating mode description in the section 7.

When this instruction is fetched, the CPU card sends to the PC the message number, used to unload the text from the file: If the text is missing, only the message number will be displayed.

5. SYSTEM FUNCTIONALITY

MAIN MENU

In this menu, all the **main system functionalities** are selectable by the user.

In this section, the different operating modes, enabled by the menu items, will be synthetically described, leaving more detailled informations to the specific sections.

In this menu level, it’s possible execute a **Shell** to the **MS-DOS** operating system, by pushing **ALT** + **D**. To return to the system, type **EXIT**.

The available operating modes, selectable by the corresponding **Function keys,** include:

F1 - ONLINE (automatic)

Allows to enter in the **Automatic**.mode: if the system has not bee initialized, the **System Initializing** mode is automaticaly recalled.

F2 - EDITOR (programs editor)

Allows to enter in the **Editor** mode for user file programming or modifying.

F3 - COMPIL (programs or functions compiler)

Allows to recall the part Programs or Functions **Compiler** , translating them in executable format.

F4 - FILE (File manager)

Allows to enter in the **File Manager** mode, where all the typical file maneging features (as Copy, Erase, Rename, Print, etc..) are provided.

F5 - CONFIG (SW configuration)

Allows to enter in the **system configuration** mode, to define all the system configuration parameters (as modules and station number, system and axes parameters, options,etc.)

F8 - LANG (language select)

Allows to load the selected language messages.

F9 - CHANGE (module change)

Allows to change the current **module**.

F10 - EXIT (exit)

Allows to exit from the OS1000 operating system, returning to the MS-DOS.

EDITOR

Preliminary

The Editor mode allows to the user to create and modify all the PTP1000 text or program files.

For the Part Program (in GPL1000 language) the **maximum text lines number** insertable in a file is **2000**.

The procedure, for Editing, is the following:

- first, the typology of the file to be edited must be selected.
- then, if new, the name and the comment must be entered, otherwise the program directory may be recalled for direct selection. The program name must be followed by the ".”character and the station number (if missing, the station 0 is assumed by default).

Function keys

During the Editor procedure, the following Function key are available:

F1 - EDIT (files types menu)

Allows to select a new file for editing: If another file is already open and modified, system requires if must be saved. In this mode it’s possible:

- modify the comment;
- rename the file: this function may be used to create a new copy of the file or to copy a file from a station to another.

F2 - SEARCH (find/substitute)

Enables the following commmands menu:

Find to search a defined characters string in the text.

Replace to sobstitute a selected string with another.

F3 - BLOCKS (text blocks selection)

Enables the following commands menu:

Mark allows to mark a group of lines of the text for a successive operation on the entire block.
The procedure is:
Place the cursor on the first line of the block and select Mark.
Place the cursor on the last line of the block and select Mark.
At this point, all the lines of the selected text block are enhanced
To disable the selected block, selected yet Mark.

Copy allows to copy one or more lines of the text.

Move allows to deplace a block of the text

Import allows to include another text file in the current file.

Print allows printing of the entire or a part of the current text.

F4 - DELETE (erase lines of the text)

Allows erasing of one or more current text lines.

F6 - TSIMB (Functions symbols table)

Allows to display the TSIMB file, including all the used symbols with the corresponding parameters.

F7 - FUN (Functions directory)

Recalls for displaying all the user Functions, relative to the current Station, saved in the different files: this command is enabled only for the A, S, F, D and E file typologies.
If the current file is of A, S, F typology, it’s possible to select and insert, in the cursor pointed position, the name of the function i the text.

F8 - OPTION (options)

Allows to select some options available in the Edit procedure: these options must be considered enabled when the corresponding menu items are marked by the character •.

These options include:

Syntax enables/disables the syntactis analysis on the text lines, concerning
only the files with **S**, **A**, **F** typology.

Read Only enables/disables the access mode to the text.
If the file is in **Read Only** status, no modifying is accepted,
avoiding incorrect alterations.
In some cases and for some files typology, this option is
automaticaly enabled by the Editor program.

Tab allows to define the different positions, along the text line, where
the cursor will be positionned when the **TAB** and **SHIFT**+**TAB** keys
are typed.

Function keys assignement allows to assign to the **function keys 1÷10**
the required mnemonic item.
This option may be very useful if these keys are assigned with the
most frequent instructions. These instructions may be inserted in the text
by typing, in contemporary, the **ALT key** + the corresponding
Function keys, avoiding to rewrite its text.

Ignore uppercase/lowercase enables/disables, during the FIND command,
the distinction, in the text, enter the upper or lower case letter.

F9 - CHANGE (change module)

Allows to change the **active Module.**

F10 - EXIT (exit from edit)

The exit mode from Editor provides the following possibilities:

Exit allows to exit from Editor
If the text has been modified, before to exit, the same feature discussed for the function key F1 - EDIT is available.
If the **Syntax** option is enabled and the file has S, A, F and D typology, the file will be automaticaly **compiled** before saving on the disk.

Save allows to save the text without exiting: as already described, the name and the comment are required.

Previous if, before the current text, another text has been edited, this command allows to recall them automaticaly, without needing selection.
If the current text has been modified, before loading the previous, features concerning the F1 - EDIT description, are enabled.

Available keys

During the text editing, the following keys are available:

ALT	enables the softkeys
TAB	next tab
SHIFT+TAB	previous tab
SHIFT+cursor up	to select, in up direction, lines of text starting from the cursor pointed line
SHIFT+cursor down	to select, in down direction, lines of text starting from the cursor pointed line
SHIFT+PAGE UP	to select previous page lines, starting from the pointed line
SHIFT+PAGE DOWN	to select next page lines, starting from the pointed line
SHIFT+HOME	to select text lines, from the pointed line to the *first* line
SHIFT+END	to select text lines, from the pointed line to the *last* line

cursors to move cursor into the rows and columns of the text

PAGE UP and **PAGE DOWN** to change the current page

HOME to select the *first* line

END to select the *last* line

ENTER to move to the first character of the next line: if INS is active, a new line is inserted.

INS to insert other characters in the line: to disable this function, type yet INS

DEL to erase characters from the cursor position towards right.

BACK SPACE to erase characters from the cursor position towards left, shifting back the rest of the line

ESC to abort the line modifying, restoring the previous text

CTRL+ENTER if INS is active, inserts a new avoid line up the current: on the contrary move the cursor on the first character of the previous line

CTRL+HOME moves the cursor to the first line of the text

CTRL+END moves the cursor to the last line of the text

CTRL+Y to delete the current line

CTRL+T to delete an entire word or a spaces sequence: the text is shifted left according to the deleted characters number

CTRL+F to repeat the string *search*

CTRL+R to substitute the pointed string with the new one (used together with the previous command)

CTRL+U to restore the last cancelled line: the line is inserted up the pointed line

CTRL+cursor right to move the cursor to the beginning of the next word

CTRL+cursor left to move the cursor to the beginning of the previous or the current word (if the cursor is positionned in the middle of the word)

CTRL+cursor up to scroll up the text without moving the cursor

CTRL+cursor down to scroll down the text without moving the cursor

PROGRAMS and FUNCTIONS COMPILING

The Setpoint and Automatic programs (filetype S e A) and Functions (filetype F) compiling allows to generate the corresponding executable (object) file to be transmitted to the Numerical Control cards (PTP200N and PLC200) for execution.

The **compiled file** is named as the corresponding source file, with added the surfix “C” character.

The *Compiling procedure is automaticaly recalled* (if the **Syntax** option is enabled) by the Editor, before saving the source file on the disk.

In the source programs **directory,** the corresponding Compiled file presence is displayed by the mark (*) in the **C column.**

Also the **I/O Definitions files** (filetype D) must be compiled, but this procedure is automatically recalled by the system, during other source files compiling.

Usually, then, the Compiling program must not be recalled specifically by the user. Nevertheless may need, for instance when some parameters are modified, to compile a lot of files.

In this case, first, user must select, in the files typology directory of the current module, the files to be compiled defining also if the listfile must be **stored on the disk or printed.**

To select the different files to be compiled, use the **CTRL** with the [+] **keys** , to deselect the **CTRL** with [-] keys.

The generated listing is stored in the **LSTCMP.R** file, where only the last compiled file is included.

The listing shows, for every instruction:

- address
- object code
- line number
- source text
- comment
- error message (if occurs)

The listing may be displayed or printed by the VIEW or PRINT commands, in the Programs Manager operatind mode or displayed in Editor; it must be selected enter the REPORT files typology.

The Listing may be very useful for debugging with the **Monitor** operation mode.

During compiling, may occurs that en error in a line generates another error message in another line, also if this is correct, as in this exemple:

```
1               PROG
 |              |
6               BRA      TENSON
 |              |
12 TENSON:      SNZ      STOP
 |              |
25              END
```

If the STOP label is missing, a first “invalid field” error is generated in the line 12.

When the Compiler matches the labels, a second “label not found” error will be detected in the line 6, because this label is refered to an invalid instruction.

When the first error will be correct, also the second error will be erased.

The Compiler menu includes the following **function keys**:

F1 - COMPIL (compiling)

Shows the directory of the files, for the active module, to allows selection for compiling.
At the end of the compiling procedure, if some errors occurs, system displays the **list** of the incorrect files.

F9 - CHANGE (change module)

Allows to change the active module.

F10 - EXIT (return to main menu)

Allows to exit from the Compiling mode, returning to the Main menu.

PROGRAMS MANAGER

This operating mode allows a lot of **Files supervision** operations concerning the user generated **Working disk**, and precisely:

F1 - DIR (directory)

Visualizes the complete **directory** of the active module files. For every file the following informations, on 7 columns, are displayed:

1) **typology** of the file (columnn **T**)
2) **name** of the file
3) **number** of the **station**
4) **comment** to the file
5) **date** of the **last modifying**
6) **lenght** in **bytes** of the **object code**; is is the number of bytes that of memory occupation in the RAM of the station card.
7) **compilation mark** is the character (*****) that signals that the object code has been generated (in the column **C**)
8) **protection mark** is the character **P** that defines that the file is protected, then cannot be modified or erased (in the column **P**)

Other two information are displayed:

- total number of files in the directory
- number of bytes free on the disk

F2 - RENAME (change name/comment to the file)

Allows to change the name or the comment in a file of the active module: First, user must select the required typology in the typefile directory.
For any file to be modified, the following informations are required:

- **title**	actual name of the file
- **name**	field where the new name must be edited
- **comment**	field where the new comment must be entered

In the case user changes the name of the program actualy in execution on the cards, returning in Automatic mode, the new name will be displayed.

F3 - COPY (copy file)

Allows to copy a lot of files from a working directory of a module to the directory of another module, or from the hard disk to the floppy disk, to produce backup copies.
In the enquery of the source and destination directories, the source directory of the active module is proposed.
If the destination directory is the disk root, as A for instance, the path **A:\** must be entered.
Then, first, user must select the file typology and, after, in the corresponding proposed list of files, select all the files to be copied.
If some of destination names is already present in the destination directory, the old file will be overwrited: if the file is protected, a second confirme is required to the user.
The configuration file cannot be copied, since not present in the directory.

F4 - DELETE (delete file)

Allows to delete one or more file of the current module.
When the file typology is selected, system provides the list of all non protected files. Then user may select one or more file to be erased, also defining if, for any erasure, a confirming request must be required by the system or not.
If the program in execution deleting is attempted, an error message is displayed and the command denied.

F5 - PRINT (print file and directory)

Allows to print one or more files, or the directory or the Functions list for the active module.

F6 - VIEW (visualize file)

Allows ti visualize the selected file.

F7 - SECURE (file protection)

Allows to protect or de-protect a file of the active module. The protected files, marked by the P character, cannot be modified or deleted.
This command is under password.

F8 - TIME (date and time change)

Displays the system date and time, allowing modifying.

F9 - CHANGE (change of module)

Allows to change the current module.

F10 - EXIT (exit)

Allows to exit from this mode, returning to the Main Menu.

CONFIGURATION

The Software configuration phase follows immediately the hardware configuration allowing to enter all the informations concerning the system logical structur in terms of number of modules, number of stations, axes number and technological features, and so on.

The system is normaly released with a defined configuration.

In any case the access to this operating mode is subordined to the password acknowledge, to avoid accidental modifying.

The availables feature are supported by the following function keys:

F1 - SYSTEM (system configuration)

Allows to enter all the general system informations, as:

• Modules Configuration

For every equipped module must be defined:

- the **Status**: defines if the module in inserted (**On**) or not (**Off**) in the system comunication loop.

- the **number of stations**: is the number (**0÷15**) of PTP200N or PLC200 cards presents in the system. If a space is entered, the corresponding module is removed from the system.

- the **Description**: is an optional description will be displayed on the screen related to the active module.
- the **Exist** field: if the (*) character is present, the corresponding module has been already configurated, otherwise it is new then its setup data must be entered.
This field is automaticaly managed by the system.

Note: when a module is remuved from the system (station number = space) its set up data saved to be restore when the module will be enabled.
If, in a module, the card number is before reduced and then restored to original value (i.e. from 3 to 2 and, after, to 3) to the new inserted cards is assigned, by default, a PTP typology.

• System options

Include general informations concerning the complete system:

- **PC COM line used**: defines the COM serial port (**1÷4**) where the PTP1000 modules are connected.

- **Baud rate**: defines the serial line Baud rate (**19200/9600**) according to the EPROM version on the cards.

- **Automatic init**: If enabled (**Yes**), at the power on, the PC begins the initializing procedure automaticaly.

- **Statistics**: defines if the customized Statistics module is installed (**Yes)**
or not in the system.

- **Number of print columns**: selects the column numbers (**80/132**) to be
used for printing.

F2 - GENMOD (module general configuration)

Once the system has been defined, this command allows the single module setting-up.
Every module may be selected by the F9 - CHANGE function key.

Data to be entered include:

• General Parameters

- **Bactery**: defines if the RAM devices include the back-up bactery, allonwing data holding also without power supplying.

- **Operation with PC**: this parameter, in connection with the previous, defines if the module works with the PC (**Yes**) or
stand- alone (**No**): in this second case is compulsory the
Bactery use.
The PC connection is, in any case, compulsory
when a multimodule architecture is used.

- **Maximum stations number**: defines the max station number equipped (**8/16**).
According this parameter, the I/O max lines
number may change, as:
1) with **max 8 stations** until 4 I/O expansions
cards may be equipped in the module.
2) with **max 16 stations** only one I/O exp.
card may be inserted.
See also the **Input/Output port numbering**
section.

- **Timeout on errors**: defines a waiting time value (**1-255** in seconds) used
in some GPL1000 instructions for errors management.

• Station configuration

For every station of a module, the following parameters must be declared:

- the **Type**: PTP if the station card is a PTP200N.
PLC type, if a PLC200.
- the **Description**: a mnemonic field may be displayed in the Console video page when the corresponding station is enabled.

- the field **Exist**: if the (*) character is present, the station is already configured: if missing, the station is new and its setup parameters must be entered.

• Working frequences

These internal data are defined by the CNC builders, then *cannot be changed* by the user without authorization, because may cause incorrect behaviour of the system.
The default values are:

- **Real time frequency**: 200 Hz.
- **Frequency of interpolation axes control:** 400 Hz.
- **Interpolation frequency**: 400 Hz. (**2000 Hz** if the **HSINT** card is provided).

• Module options

These options may be selected according to the application requirements.

- **Selection of init programs**: if enabled **(Yes)** allos to select the working programs in the system Initializing phase (see relative description.

- **Selection of init tables**: as the previous parameter, but related to the quotes Tables.

- **Common programs names for all stations**: if enabled (**Yes**), defines that all the executable programs must have the same names for every station.

- **Common tables names for all stations**: as the previous, but refered to the quotes Tables.

- **Quotas Tables**: if enabled (**Yes**) defines that the system uses the Quotes
Tables features.

- **Function with parameters** if enabled (**Yes**) defines that system uses
Parametric Functions (see also the relative descriptions).

- **Function with parameter tables**: prevue for future use, actualy not available.

- **Use of Setpoint operation**: if enabled (**Yes**) allows to perform a specific
SetPoint procedure. In the contrary, the
Setpoint procedure must be included in the
Automatic programs.

- **Use of Edit operation in automatic**: if enabled (**Yes**) allows
user to access to the Editor
operating mode also
during the Automatic mode.

- **Real Quotes Displaying**: if enabled (**Yes**) in the quotes displaying
operations, the real quotes will be
considered (not the theorical quotes).

• **Options for Diagnostics**

These options may be enabled when a REPORT file is required to monitor and store the machine status during automatic functionality.
These options include:

- **Save Cycle/System errors**: if enabled (**Yes**), during the
automatic process the system provides storing
of all the cycle and system error messages,
with the date and time, and the
module and station where the error occurs.

- **Save VIDEO messages**: if enabled (**Yes**) actives storing of
the all Video messages, with the same
format of the previous case.

- **Save Counters**: if enabled (**Yes**) actives the counters storing.

- **Counters descriptions**: if enabled (**Yes**) allows the access to the next
point

• **Counters descriptions**

This mode, enabled by the previous diagnostics options, allows to assign a description to until 32 counters, also from different cards, to be displayed in

the Automatic page: the counter, may be used as timers, are managed by dedicated GPL1000 instructions.
The data must be entered are:

- **Counter number**: 1) if a value **from 0 to 31** is entered, it is assumed as counter address
2) if **space** , all the data, related to the station and the description, are erased.

- **Station number**: number of the station where the counter is allocated

- **Description**: is the description message will be displayed together the counter value.

F3 - STAT (station configuration)

When the Module general parameters have been entered, the following data must be programmed, related to every station (with the F9-Change key any station may be selected anytimes).

• Axes parameters (DC current)

An axis is considered configurated when all its parameters have been programmed. To remove an axis, all its parameters must be erased.
These parameters include:

- **Description**: optional description field may be displayed, instead the usual axis name.

- **Resolution**: number of encoder pulses, after the electronic multiplying, per mm.
This value must be programmed according the card predisposition by the multiplier code jumpers.

- **Maximum Speed**: defines the maximum programmable speed, in mt/min.

- **Acceleration**: defines the acceleration time (in milliseconds) to reach, starting from zero, the programmed maximum speed.

- **Window**: defines the max position error (in encoder pulses) accepted by the system to consider the moving ended.

- **Gain**: defines the proportional position loop gain (value **0.25** to **15)**.
The Op Amp feedback resistor has usually 20 Kohm value, then defining a unitary Gain. Change proportionaly this value to change the Gain.

- **Positive limit**: SW limit swicth, defining the maximum programmable quote, in positive direction.

- **Negative limit**: as the previous, but in negative direction.

- **Feed forward p (point to point)**: include 2 values, separated by the / character. Allows to evaluate the feed forward contribution (proportional to the actual speed) to be added to the position error value. The value is defined in fractional format, where the first value (**0.25÷15)** is the multiplier**,** the second (**1,2,4** or **8**) the divisor.
Then: Kv = 1° value*V/2° value

- **Feed forward i (interpolation)**: This parameter must be entered as floating point number (if the HSINT card is equipped) or as fractional number (if the HSINT is missing): in this case the number will be defined as a multiplier (**1÷32**) and a divisor (**0.25÷15**).

• Axes parameters (AC current)

An AC axis is considered configurated is all its parameters (excepting the High/Low speed output line) are entered. Then, to remove an axis, all these values must be erased.
These parameters include:

- **Description**: Optional descriptional field, as in the DC axis case.

- **Resolution**: As in the DC axis case

- **Positive quote window**: defines the encoder pulses number, before the final quote position, in positive direction, where the motor is stopped.

- **Negative quote window**: as the previous, but in negative direction.

- **Forward output**: defines the output line (bit+port) used for motor driving in the positive direction.

- **Backward output**: as the previous, for negative direction.

- **High/Low speed output**: defines, if used, the change motor speed output line.
This output line will be actived (if high speed is selected) only if the programmed deplacement

is almost 4 times greater then the corresponding window value.

- **Interpolation (only for DC axes)**

These parameters relate to the HSINT card:

- **Interpolation board:** type **Yes** if the HSINT card is equipped.

- **Maximum Speed:** maximum axes speed, in interpolation (in m/min).

- **Acceleration:** acceleration time (in millisec.) from zero to max. speed.

- **Input/output expansion cards**

These parameters define the I/O expansion cards typology, enter:

- **Inoutr** fixed 24+24 In/Out
- **Iomod** supports until 6 plug of 8 inputs (Modinp) or 8 outputs (Modout)
- **Remote** supports until 16 remote I/O modules

If in the previous parameter the **Iomod** has been programmed, or the station is a **PLC200 type,** these other parameters must be entered:
For any plug equipped, the typology must be entered, enter:

Input	8 optocoupled inputs
Output	8 relais outputs
Aninp	2 analog inputs
Anout	2 analog outputs
Ac	3 channel encoder pulses counters
Remote	Rx/Tx microcontroller for remote I/O modules interfacing

Note: if the card is removed, all the relative plugs will be erased.

- **Memory areas used on board**

The total **available memory (RAM) bytes number** for the user, in the PTP200N or PLC200 card **is 27648.**
It's possible to dedicate all the memory area for the user program or segment them for different uses:

- **Programs**: memory area reserved for Set Point (if used) or Automatic programs and Functions.

- **Tables:** memory area reserved for quotes tables

- **Function Parameters:** memory area reserved for the Functions Parameters tables (actualy not implemented, then to set to zero).

- **Others:** memory area reserved, if required, for the Logic Analyzer sampled data storage.

• Special utilities Functions

These Functions, defined by her number or name, will be executed when one of the following conditions occur:

- System Errors
- Cycle End
- Emergency

The **PLC function** is started at the end of the initializing procedure, after the power up, staying active until the power down.

Note: The Special purposes functions must be entered, and the corresponding file compiled, before to declare them in the Configuration.

• Emergency Table

In this table must be declared the cards input lines where emergency signals are cabled (as Field Stop, Limit switches, etc...).

System provides until 16 emergency lines management: for everyone a Firwmare subroutine is scheduled when the corresponding input line switches on the active logical status. The input line must return in the quiescent status so that the emergency control has been enabled again.
The following data must be entered:

- **Input**: address (as bit+port or in symbolic format) of the input line where the emergency signal is connected.

- **Status**: defines the normal (quiescent) logical status of the signal:
  Nc if normaly *closed*
  No if normaly *open*

In the PTP1000 standard version, some emergency conditions are provided by the system: to every condition is associed a **System Error** which message is displayed in the **Description** field.

These emergency conditions include:

1. **general emergency:** stops the axes movements and ends the program. If enabled, the special Errors management function is started.

2. **axis X - emergency:** as the emergency 1.
3. **axis Y - emergency:** as the emergency 1.
4. **axis Z - emergency:** as the emergency 1.

5. **axis W - emergency:** as the emergency 1.
6. **axis V - emergency:** as the emergency 1.
7. **axis X - zero limit switch:** as the emergency 1.
8. **axis Y - zero limit switch:** as the emergency 1.
9. **axis Z - zero limit switch:** as the emergency 1.
10. **axis W - zero limit switch:** as the emergency 1.
11. **axis V - zero limit switch:** as the emergency 1.

12. **auxiliary emergency:** no System Error is generated, the program dont stop: if enabled, the Emergency special function is runned.

13. **auxiliary emergency:** as the emergency 12.
14. **auxiliary emergency:** as the emergency 12.

15. **Field stop :** stops the program execution and the axes movement. It's equivalent to the STOP command from PC.

16. **Field start:** starts a program suspended by a previous STOP command: axes restart towards the original target position.

Note about the emergency 7÷11: these emergency conditions are detected only if the SPEX (bit+porto 2003) flag is = 1 (see also the **Input/Output ports numbering**).
In this mode, the same zero limit switch may be used for the Setpoint procedure and, after, as emergency.

- **Feed rate override (only for DC current motors)**

This parameter relates the input port address (001÷255), or the corresponding symbolic definition, where the Feed Rate Override potentiometer, if used, is connected. The FRO device allows the user to modify the current interpolation speed, in the 0.4% to 100% range.
The potentiometer may be connected to the input line of a MODAINP plug or to the ESPAS analog input: in this last case, the port address is 255.

- **Portable Keyboard Parameters**

If the CAT90 portable keyboard is used, the following parameters must be entered:

- **Function**: is the number (1÷255) or the symbolic name of the function used for keyboard management.

- **First port**: is the address of the first of the 8 contiguous flags ports used for the keyboard management.

F4 - PRINT (Print Configurations)

Allows to print the Setup data in the 4 different modes.

- Complete configuration: allows to print the complete configuration files, concerning:
the System
all the modules
all the stations of every module

Printing may be addressed to the Printer or to the disk (in this case the CONFIG.PRN file in the system directory).

- System configuration: prints the data concerning the F1 - SYSTEM description.

- Active module configuration: prints all the data concerning the active module and the corresponding stations.

- Active station configuration: prints the Setup data (see F3 - STAT description) of the active station.

F9 - CHANGE (change module/station)

Allows to **change** the current **module** and **station** .

F10 - EXIT (exit)

Allows to return to the Main Menu.

PARAMETERS RETRANSMISSION TO THE STATIONS

If, when the system has been already initialized, some parameters are modified, as the Automatic mode is recalled, an automatic parameters retransmission, to the involved stations, is performed.

Since the new data retransmission *dont stop* the program in execution, user takes care in the parameters modifying, avoiding them, if possible, during a program execution. User overall must remember, for instance, that the **axes resolution** or **speed** modifying requires the programs and functions re-compilation.

Obviously, changes of the system configuration, concerning, for instance, the modules and stations layout or the axes number, require a new system initializing.

7. MACHINE MANAGER

START UP

The System Start up must follow the sequence:

- PTP1000 power on
- the Personal Computer power on

If the **Option : Automatic Initializing,** in the **System Setup,** is enabled, the PC actives the connection sequence to the electronical modules and, in the case of the first connection, send to every station, in the order:

- the Parameters
- the Functions (if present)
- the Setpoint Program (if used)
- the Automatic Program
- the Table of quotes (if used)

During transmission, the system displays a warning messages if some transmitting files have not be found.[1]

If the connection has failed, or not all the files have been transmitted succesfully or the Automatic Initializing option is disabled, PC enters in the **System Initializing** operating mode.

If a PC is connected to a already initialized system, the automatic connection, indipendently of the Automatic initializing option is attempted and, if failed, system enters in the System Initializing mode.

SYSTEM INITIALIZING

This operating mode manages the system start up under user control, to complete the PC to Modules logical connection.

The following functionalities are available:

[1] . If the cards are equipped by the bactery CMOS RAM, the PC sends a checksum status enquiry: if correct, no transmission is performed, to avoid, in large systems, a lost of initializing time.

F1 - INIT (initializing)

Allows to start the initializing procedure.
May be used, i. e., when the electronic module is powered after the PC.

F2 - CHGPRG (program change)

Allows to change the programs to be transmitted, selecting them enter the compiled files directory, related to the current module. Then, edventualy, user must recall the **Module Change (F9 - CHANGE)** to select the required one.

F3 - CHGTAB (tables change)

If in the current module the Tables of Quotes functionality is provided, this command allows to select the Tables file to be transmitted to the required stations. The Module Change command (F9 - CHANGE) allows to select the required Module.

F4 - RETRY (total re-transmission)

This command allows to transmit all the data to the stations of all the modules.

F5 - AUTO (automatic)

This command, enabled only if the system is already initialized, allows the re-entry in the Automatic operating mode.

F9 - CHANGE (module change)

Allows to select the module where the **Program change (F2 - CHGPRG)** or the **Tables change (F3 - CHGTAB)** commands are required.
The last selected module, before the **Initializing (F1 - INIT)** starting, will be the current module when entering in the **Automatic** mode.

F10 - MAIN (main menu)

Allows to enter in the System Main Menu.

AUTOMATIC MODE

The Automatic operating mode allows to control and monitor the machines working cycles.

Some function keys provide a direct commands over the machine cycles execution, other allows to recall some accessory functionalities.

In the Automatic mode, PC polls periodically all the hardware modules to check, for every one, his functioning status, then allowing to detect errors or alarm situation and to receive any type of warning messages.

On the screen are supplied:

- a box including the axes quotes, related to the current station (in the case of AC axes, 6 quotes at once are displayed).

- a box including informations about the executing programs, as the name, the comment and the Table of Quote in execution (if used), related to the current station.

- the messages sended by the VIDEO and MESSAGE instructions, inserted in the part programs (on the screen a max of 8 lines are provided for displaying):

  the VIDEO messages are alwais displayed on the row n. 8

  the MESSAGE instruction string may be addressed on the row from 1 to 8, as
  defined in the instruction code.

- the Cycle or System Error messages

- the Counters and timers contents, only if enabled.

The **function keys** of the operating mode include:

F1 - START (execution start)

Starts execution for the Automatic Programs, for all stations of all the modules.
After a STOP, the START key restarts the programs from the suspended line.
The START command is accepted only if all the selected stations programs has been transmitted to the modules CPU cards.

F2 - STOP (execution suspend)

Suspends all the programs execution, stopping the axes with slowing down.
Programs may be restart by the START command.

F3 - END (execution end)

Stops definitively the programs execution.
The START command restart the programs from the beginning.

F4 - LOCAL (local mode)

If enabled, the START, STOP and END commands operate only to the current module.
To disable, repeat LOCAL key pushing.

F5 - SETP (setpoint)

This finction key is present only if, in **Set up** mode, the **Setpoint Mode Option** has been enabled: this command start the Setpoint procedure, with the following subcommand keys:

F1 - START (setpoint start)

Runs the Setpoint program on all the stations of all the modules, checking if in all the enabled cards this procedure has been transmitted.
If the Setpoint procedure ends successfully, without System or Cycle errors and if hasn't suspended by the F3-end key, system returns in the Automatic mode.

F3 - END (setpoint end)

Ends the setpoint program for all the stations. The START key restart the procedures from the beginning.

F4 - LOCAL (local mode)

If enabled, the START and END commands are sended only to the current module.
Repeat selection to disable.

F9 - CHANGE (module/station change)

Allows to change the current Station and Module.

F10 - EXIT (return to automatic mode)

Allows to return to the Automatic operating mode.
If some Setpoint procedure is already in execution, this command is denied.

F8 - MENU (accessory operating modes)

This menu include the following accessory operating modes:

Manual and diagnostics
Re-transmissions
Programming
Initializing system
Edit

discussed later.

F9 - CHANGE (module/station change)

Allows to select another station or module.

F10 - MAIN (main menu)

Allows to exit from Automatic mode, returning to the Main menu.

MANUAL AND DIAGNOSTICS

This operating mode may be used for **axes manual moving**, the **Input/Output lines control** and to start a **non parametric functions execution,** related to the current station.

Entering in this mode DONT stop program execution of the current station, then user MUST TAKE CURE to avoid dangerous operations.

On the screen two main boxes are displayed: the first dedicated to the moving axes, the other to the digital I/O signals:

- **Moving axes box:** this box includes, for the enabled axes, the following informations:

  the name
  the description
  the quote
  the current status

  One of the axes is enhanced as current, showing that to this the **Pitch** and **Speed** informations are refered (only Pitch for AC axes).

  To change the axis, user may type the corresponding letter, recalling the new axis informations; for AC axes, also the F4 - AXIS key may be used.

  To move the selected axis, type the + or - key, according to the required direction: moving will be executed according to the selected parameters (status, pitch, speed).

- **Input/Output box:** in this box the selected card I/O lines are displayed: any column relates to an Input or Output port, the 8 rows relates the single line (from 0 to 7).

  In the box, 3 ports are showed at the time, then a total of 24 I/O bit; the line (bit) address may be obtained adding the port number with the row position.

  Any line include an 8 character field showing the mnemonic definition, if programmed in the Edit mode (otherwise a void field will be presented).

  The line status (active/disactive or on/off) is showed by the enhancing bar.

The available **function keys** include:

F1 - JOG/ST (Jog/Step axis moving mode)

Toggle key to change the JOG or STEP mode.
In the **JOG** mode, the axis move begins when the (+) or (-) moving key is pushed, and ends when released.
In the **STEP** mode, an incremental displacement, according to the Pitch parameter, is performed. To **stop** the axis, type the **Space bar** key.

F2 - FREE (axis control loop disable)

Valid only for DC axes.
When typed (toggle key), the axis status change enter the **FREE status** and the previous (jog or step).
When an axis is in FREE status, no moving commands are possible: axis may be mechanically moved, maintaining the quotes counting actived.

F3 - AXPAR (axis parameters)

Allows to insert or modify the **pitch** and **speed** parameters of the current axis, only if the axis is not in Free status. For AC axes, only the Pitch parameter may be programmed.

F4 - AXIS (axis selection)

Allows to select the current axis, alternatively to the direct selection by the phisical names. For the AC axes, this commend switches to a following 6 axes group.
Only the axes box is modified by this command, unchanging the I/O box.

F6 - IN/OUT (I/O selection and test)

This command allows to change the I/O port group in displaying, with automatic updating of the corresponding informations.
Pressing the Space Bar, is possible to active/disactive the selected output line or Flag bit.
Exiting, the last adressed ports are leaved selected also during other operation, excepting a new station or module selection (see F9 - CHANGE).
On side of the port address, lower-case letter **'i'** or **'o'** shows the input or output typology.

F7 - FUN (function execution)

This command allows to start in execution immediately the selected non-parametric function, enter the numbered or mnemonic list.
During the Function execution, all the different boxes informations are updated: the Function execution may be stopped by the **Space Bar** key.

F8 - VOUT (analog output drive)

Allows to set a voltage value (enter -10.0 to +10.0 volt) on the 6 analog output.
Valid only for DC axes.

F9 - CHANGE (module/station change)

Allows to select a new station or module as current. When changed, the axes and I/O boxes are updated.

F10 - EXIT (return to Automatic mode)

Allows to exit from the operating mode, returning to the Automatic mode.

PROGRAMMING

The **Programming** mode allows to create or modify a user program, to execute a single line or a block of statements and to autolearn the axes quotes, using guided and interactive procedures.

This operating mode is available only in a special version of the PTP1000 system and is discussed in the Sect. 8.

RETRANSMISSIONS[2]

This mode allows to retransmit to the current module the complete set of Programs, Functions and Tables of Quotes.

May be used if user need to change the working programs on some stations, without re-initializing all the system.

May be also used to send to the card a modified program.

SYSTEM INITIALIZING

This is the operating mode described in the beginning of the section.

This mode may be used when the hardware module has been cutoff and,after, powered without powering off the PC.

EDIT

This Menu item is available only if, in the **Setup** mode, the **Automatic Edit Option** has been enabled.

Allows to enter in the **Editor** mode without crossing from the Main Menu.

[2] . Re-transmission causes the executing programs ending. In the case of Functions re-transmission, also the Automatic and Setpoint programs are transmitted, according to the memory layout on the board.

Editor loads automatically the current station executing working program and tables of quotes (if used), allowing modifying.

Exiting from Editor, the system return automatically to the Automatic mode, immediately re-transmitting the modified programs, functions and tables of quotes.

APPENDIX A

System errors

Introduction

System errors are automatically detected by the cards and sended to the PC for displaying.

These errors relate different origins: axes failures, communications problems, etc. When a system error occurs, the corresponding module programs execution stops.

Every error is displayed with a number code, to simplify international service.

Following all the error typologies are listed, including an explication about the possible causes.

EMERGENCY INPUT LINES ALARMS

General Emergency

Defined in the Emergency table[1], normally is used to advise for emergency push-button pressing, for axes limit-switches and/or or other signals must stop immediately the axes movements.

Auxiliary Emergency

Defined in the Emergency table, may be used to start the emergency management Function[2].

[1]. See also the section related to the “Emergency tables”in the Station Configuration.

[2]. See also the section related to the “Emergency management”in the Station Configuration.

AXES ALARMS

All the error conditions related to the axes, if not expressially indicated, stop the axes travel resetting the reference signal and disabling the control loop for one second to avoid overshoot.

Axis ... - emergency

Defined in the Emergency table[3], is used for immediate axes stop when the limit switch of emergency is detected.

Axis ... - zero limit switch

Defined in the Emergency table, is used for immediate axes stop when the limit switch of emergency, used also as zero reference, is detected.
. This emergency then is conditioned by the SPEX flag: if SPEX=0 this alarm is ignored, allowing the zero point search procedure.

Axis ... - incorrect encoder connection

This error is detected when, with axis quiescent, a loop error greater than 256 encoder pulses is detected. In this case, the reference signal is resetted to 0 voltage and the axis placed in the FREE status.

Axis ... - not enabled

This error is detected when a point to point axis moving is recalled but the axis is not enabled for this moving, because in interpolation or in coordinated movement,etc.

Axis ... - not ended movement (for DC axis)

This error is detected at the end of a displacement, if after 5 second from the end of the theorical movement, the loop error (as difference enter the theorical and real axis quote) is greater of the threshold defined in the Setup mode. This error may be caused by a wrong offset regulation of the reference analog output on the card or on the servo drive. May also be caused by a mechanical clearance or by a position loop gain too low.

[3]. See also the section related to the “Emergency tables”in the Station Configuration.

Axis ... - not ended movement (for AC axis)

This eerror is detected if, 5 second after the stop command, system detects an axis movement by the encoder channel counting.

Axis ... - servo error (DC axis)

This error is detected when, during an axis movement, the loop error of position overcome the 2047 encoder pulses limit. This may be caused by a wrong regulation of the position loop gain (on the card or on the servodrive) or by mechanical interferences or eccessive inertia.
User is suggested to verify the correct functionning of the encoder and the set servodrive/DC motor, using the Monitor commands.

Axis ... - servo error (AC axis)

This error is displayed when, after one second from the start command, no axis displacement is detected.
Verify the encoder connection and the outpu command lines to the motor.

Axis ... - hors positive limit

This error is displayed when the theorical quote overcomes the programmed (in Setup) positive limit.

Axis ... - hors negative limit

As the previous, but in negative direction.

ERRORS RELATED TO THE MEMORY LAYOUT

Memory function full
Memory immediate programs full
Memory parameters tables full

These errors are detected when the corresponding memory areas have not enough capacity to receive all the data transmitted by the PC. To avoid this problem, user must modify, in the station Setup, the corresponding areas dimensions.

ERRORS RELATED TO CONFIGURATION

Axes expansion card not present

This error is detected if, missing the ESPAS card, a Z or W or V axes moving instruction is attempted.

Interpolation card not present

This error is detected when, missing the HSINT card, the corresponding parameters are sended to the CPU card during the station setup.

Serial I/O module not present

This error is detected when, missing the remote I/O controller card, the corresponding parameters are sended to the CPU card during the station setup.

ERRORS RELATED TO THE PROGRAMS EXECUTION

Function not found

The FCALL instruction recalls a non existing function.

Function already in execution

This error is detected in the case of recursive nesting of function: for instance, the main program performs a FCALL 10, this including an FCALL 20 instruction, this last including an FCALL 10 statement.

Too many nested functions

This error is detected when the function nesting number is greater than 4.

Instruction FRET not recalled by FCALL

This error is detected when a FRET instruction, without a previous FCALL statement, is attempted.

Function parameters incorrect
Function parameters table not found

These errors are detected when the bytes number of parameters, required by a function, dont match whith the number inserted by the calling program or defined in the parameters table.

Pointer for Table of quotes not initialized
Parameters for table of quotes incorrect

These errors are detected if an icooerct tables of quotes use occurs. Usually are detected when some instructions recall a table of quotes not already inserted.

Table of quotes index incorrect

This error are detected if the moving instruction use an index pointer greater than the maximum length of the selected table.

Too many nested subroutines

The number of nesting for subroutine is greater than 4.

RET instruction not recalled by CALL

This error is detected when a RET instruction, without the corresponding CALL, is attempted.

Illegal instruction op. code

This error is detected when an incorrect op. code instruction is fetched: user must control the op. code list in the GPL1000.TPA file (included in the OS1000 operating system).

Axis illegal mode

This error is detected in case of incorrect use of instructions modifying the axis status: for instance, if, during an interpolation or in CHAIN movement, a FREE instruction is fetched.

Incorrect axis offset pointer

This error is detected when an instruction recalls an axis offset number hors the permitted range.

Incorrect synchronism parameters

This error is detected when a SYNC or WSYNC instruction relates not existing station.

Too many nested repeat

This error is detected when the nested REPEAT number is greater than 4.

ENDREP instruction without REPEAT

This error is detected when an ENDREP instruction, without the corresponding REPEAT, is fetched.

ERRORS RELATED TO INITIALIZING

Program not present in the directory

This error is detected when a not transmitted program is attempted for execution.

Card parallel interface error

This error may be caused by an incorrect addressing, by jumpers, of the slave cards or by a parallel interface failure.

Serial I/O module error

This error is displayed when a failure, enter the PTP200N or PLC200 cards and the Serial I/O module or remote I/O controller, is detected. User is suggested to check the optical fiber cable and the remote power supply.

APPENDIX B

Analyser

GENERAL DESCRIPTION

The analyser functionality allows storing and displaying in graphic form the time behavior of physical entities controlled by the system. This functionality may be accessed in **Automatic** mode only, while within Program Management functionality the recorded samples may be displayed:

The analyser samples one or more variables at selected time intervals; it works off-line, so it doesn't allow displaying the sampled signals in real time, but it stores data that can be analyzed later.

Analyser operation is constituted by two main blocks: definition of data to be sampled and stored, and graphic display of sampled data.

Data storage uses memory area named "others", so that sufficient space in this area must be provided [1].

The more space is configured in this area, the higher will be the number of data sampled, as the analyser will go on storing data until the assigned memory area is full.

In order to start a track, the operator must define:

- basic parameters
- triggers
- data to be sampled.

Access to these data is provided by function key F5-EDIT.

Data display is provided by function key F6-VIEW.

Data **Bitport** and **Counter Number** may also be set by using the corresponding symbolic definition.

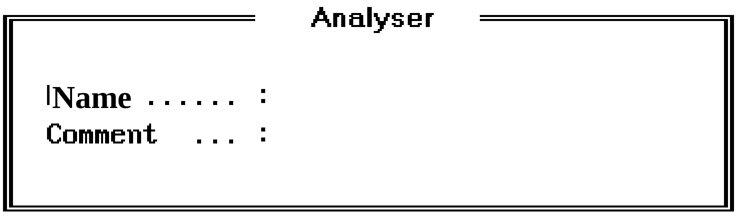
[1] . See **Memory areas used on board** in Station Configuration.

ENTERING ANALYSER OPERATION

Analyser functionality is activated in **Automatic,** by pushing keys **ALT+O**, even while automatic program is running.

The analyser works on the active station, but station can be changed by pressing function key **F9-CHANGE**.

The following box will be displayed:

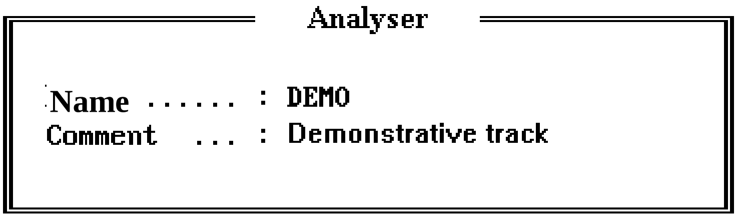


Analyser

Name ...... :
Comment ... :

Creation of a new track

For creation of a new track, digitize filename and, if desired, a comment, as indicated in the box below:



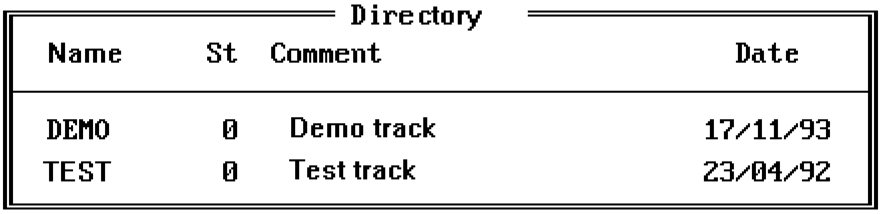
Analyser

Name ...... : DEMO
Comment ... : Demonstrative track

In this case, the system will create a new file containing both sampled data and parameters corresponding to the track.. Data will be stored after sampling pressing function key **F3-STORE**.

Selection of an existing track

For working on an existing track, press key **ENTER** in the first box; the screen will display a list of the existing files in the active station:



Directory

Name	St	Comment	Date
DEMO	0	Demo track	17/11/93
TEST	0	Test track	23/04/92

Select a file and press **ENTER**. All data used for that track will appear on the screen; You can modify them by pressing function key **F5-EDIT** in order to access to Edit menu.

DESCRIPTION OF FUNCTION KEYS

Function Keys available are the following:

F1 - START Verifies trigger and starts sampling; on the top right corner the screen will display the message: "sampling in progress".

F2 - END Exits trigger verification or data sampling before completing track; all data sampled will be lost

F3 - STORE Stores track in a file

F4 - FILE Allows editing a new file or displaying an existing file. The procedure is the same as described for entering analyser operation.

F5 - EDIT Allows editing parameters.

F6 - VIEW Displays track data in graphic form.

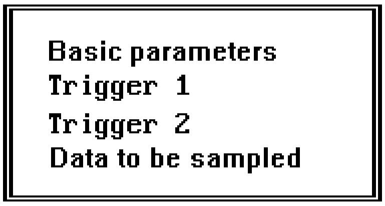
F9 - CHANGE Allows sampling in a different station.

F10 - EXIT Exits analyser operation.

Almost all functions associated to function keys are self explaining; a more detailed explanation is given hereafter for **F6-EDIT** and **F7-VIEW**.

F5-EDIT

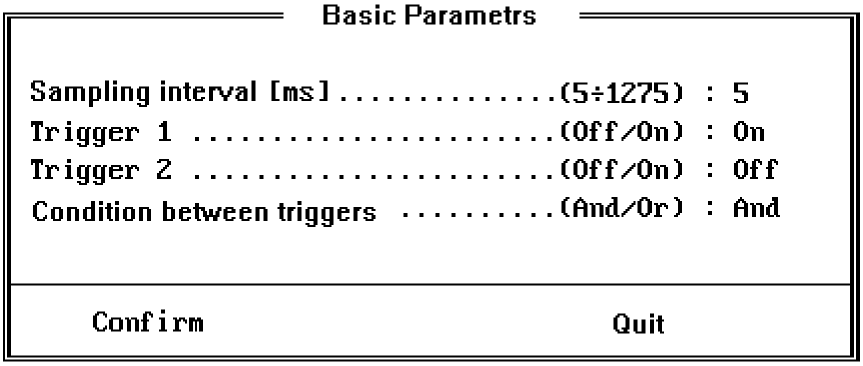
Edit operation allows introducing or modifying **basic parameters**, **triggers** and **data to be sampled**. The menu is the following:



Basic Parameters

Basic Parameters allow configuring the analyser. Specifically, they define sampling interval, triggers (one or two) and logical correlation between triggers: **OR** if sampling should start as soon as one of the two triggers is verified, **AND** if both of them should be verified.

The following box will be displayed:



```
Sampling interval [ms] .............(5÷1275) : 5
Trigger 1 ......................(Off/On) : On
Trigger 2 ......................(Off/On) : Off
Condition between triggers ..........(And/Or) : And
```

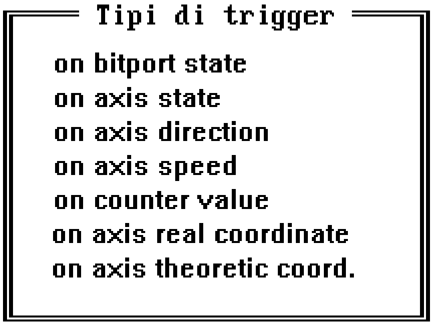
Sampling Interval: Time in milliseconds between each sampling; minimum time corresponds to real time period[2].

Trigger 1 and 2: Defines trigger state: On or Off. When Off, trigger is not considered.

Condition between triggers: Defines the logical condition between triggers.

Trigger 1 and 2

Triggers are used for starting storage; various types of triggers may be selected within the following menu:

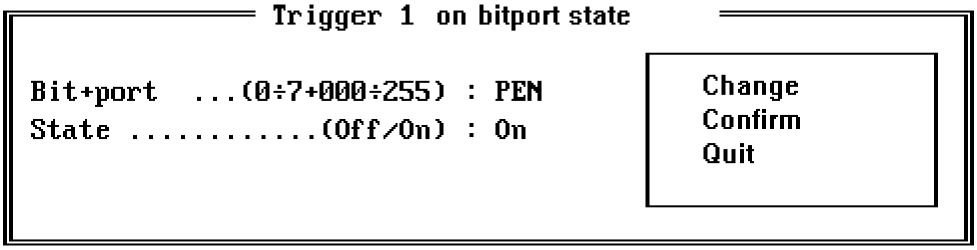


Each item and the corresponding box are described hereafter.

[2]. See **Working frequencyes** in General module Configuration.

1) on bitport state: storage begins when the selected bitport is in the state defined;

inputs, outputs and flags may be selected.



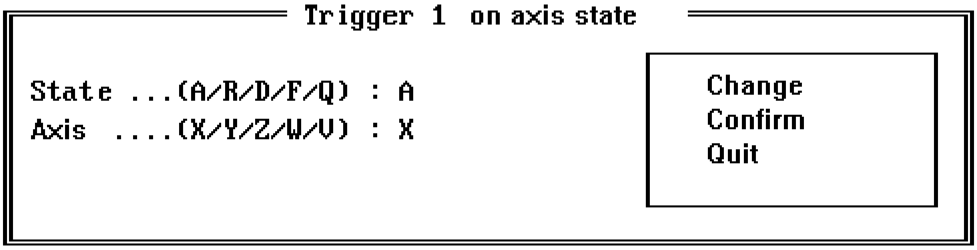
Trigger 1 on bitport state

```
Bit+port  ...(0÷7+000÷255) : PEN
State ............(Off/On) : On
```

2) on axis state: storage starts when axis is in the selected state, axes may be tested in

the following states:

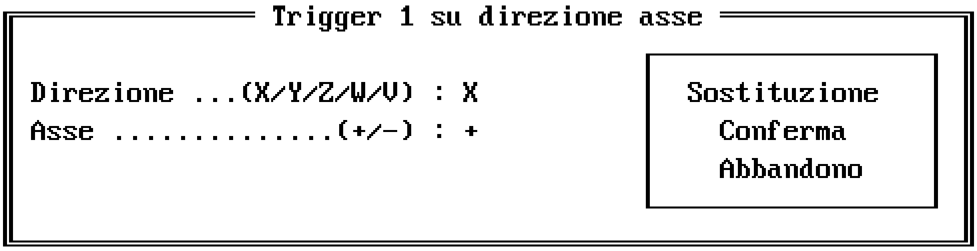
A	Acceleration
R	Regime
D	Deceleration
F	Theoretic movement completed
Q	Axis in position



Trigger 1 on axis state

```
State ...(A/R/D/F/Q) : A
Axis  ....(X/Y/Z/W/V) : X
```

3) on axis direction: storage starts when axis is moving or beginning to move in the selected direction; axes can be tested in positive (+) and negative (-) direction.

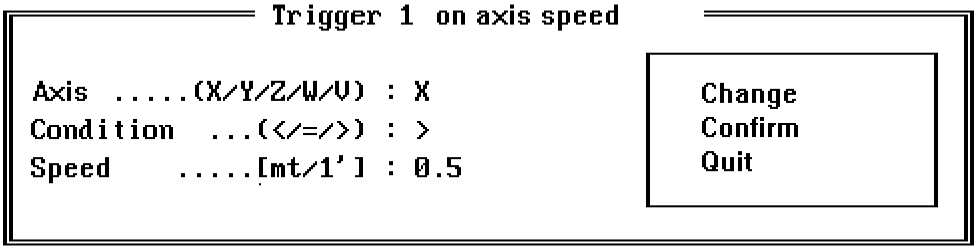


Trigger 1 su direzione asse

```
Direzione ...(X/Y/Z/W/V) : X
Asse .............(+/-) : +
```

4) on axis speed: storage begins when axis runs at a speed corresponding to the condition defined, with reference to programmed speed; axes may be tested with respect to the following conditions:

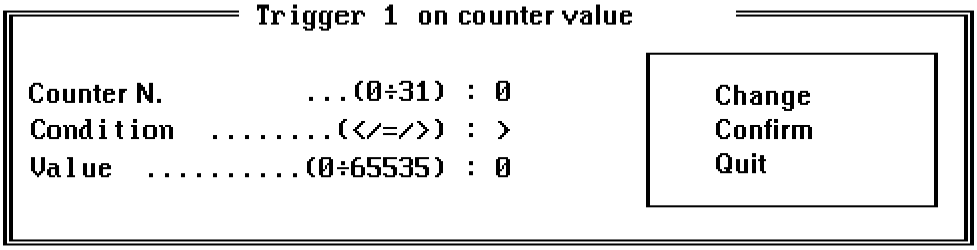
<	axis speed lower than indicated
=	axis speed equal to indicated
>	axis speed greater than indicated



```
Trigger 1  on axis speed

Axis  .....(X/Y/Z/W/V) : X
Condition   ...(</=/>) : >
Speed    .....[mt/1'] : 0.5
```

5) on counter value: storage begins when the selected counter contains a value that satisfies the condition indicated, with respect to the programmed number:

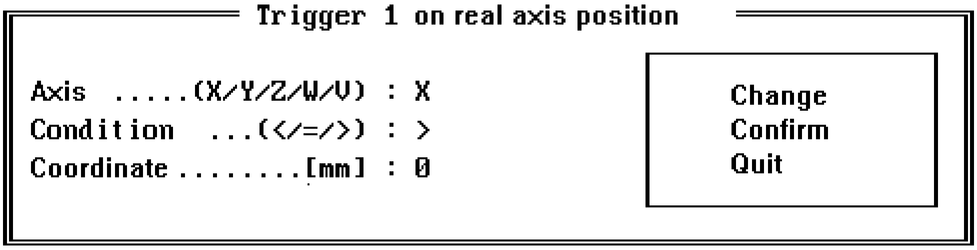
<	counter lower than programmed number
=	counter equal to programmed number
>	counter greater than programmed



```
Trigger 1  on counter value

Counter N.          ...(0÷31) : 0
Condition   ........(</=/>) : >
Value  ..........(0÷65535) : 0
```

6) on real axis coordinate: storage begins when real position of selected axis satisfies the condition indicated:

<	axis coordinate lower than programmed
=	axis coordinate equal to programmed
>	axis coordinate greater than programmed

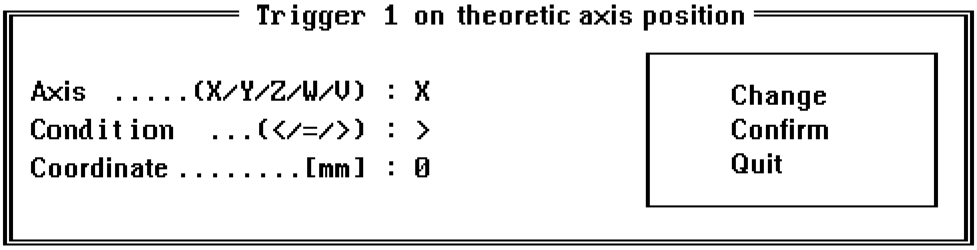


Trigger 1 on real axis position

```
Axis  .....(X/Y/Z/W/V) : X
Condition  ...(</=/>) : >
Coordinate ........[mm] : 0
```

7) on theoretic axis position: storage begins when theoretic position of selected axis satisfies the condition indicated:

<	axis coordinate lower than programmed
=	axis coordinate equal to programmed
>	axis coordinate greater than programmed



```
Trigger 1 on theoretic axis position

Axis  .....(X/Y/Z/W/V) : X
Condition  ...(</=/>)  : >
Coordinate ........[mm] : 0
```

Change of a trigger type

For changing a type of trigger, select **Change** within the box; the menu of available types of trigger will be displayed.

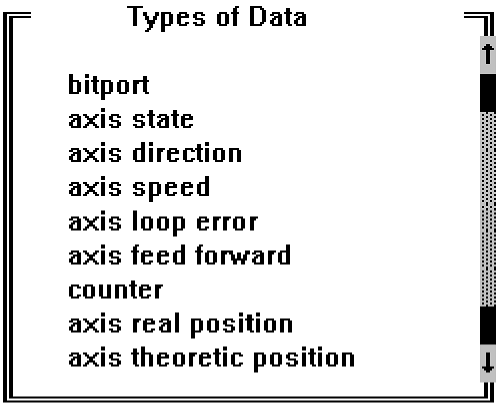
Data to be sampled

In this section one may define all data that shall be stored after trigger; the more data one selects, the lower will be the number of samples that may be stored.

Each data may occupy 1 or 2 or 4 bytes in memory; specifically:

- 1 byte for: bitport, axis state, axis direction.
- 2 bytes for: axis speed, axis loop error, axis feed forward, counter.
- 4 bytes for: axis real position, axis theoretic position

The following types of data are available:



Types of Data

bitport
axis state
axis direction
axis speed
axis loop error
axis feed forward
counter
axis real position
axis theoretic position

A maximum of 16 data may be sampled. Within box, data is selected by means of **cursor keys**, **TAB** key allows selecting the button for the function desired, that is activated by **ENTER**.

The buttons provide the following functions:

Change Allows substituting current data with a new one.
Under the box, the menu of Types of Data is displayed, for selection of the new type.

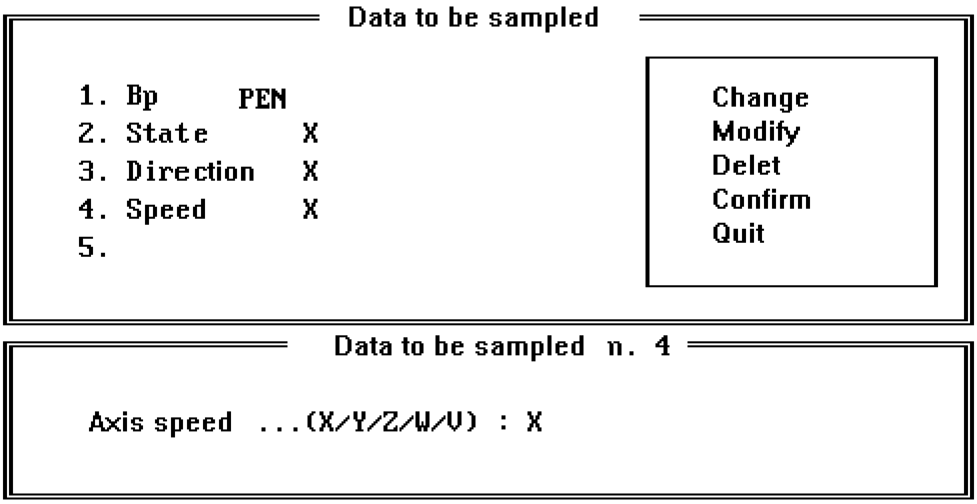
Modify Allows modifying current data with another data.
The box containing the preceding data is displayed, so that it can be modified.

Delete Deletes the selected data.

Confirm Confirm modifications and terminates editing of Data to be sampled

Quit Exits without saving modifications.

The following boxes show how data number 4 may be **Modified**:



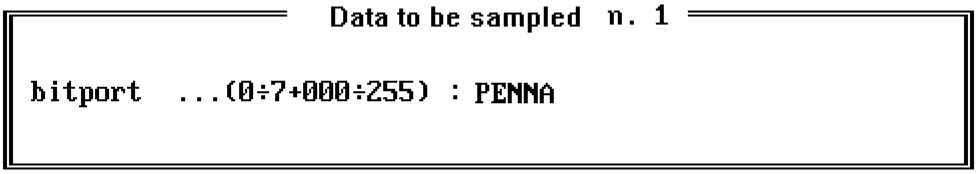
```
Data to be sampled

1. Bp       PEN
2. State       X
3. Direction   X
4. Speed       X
5.
```

```
Data to be sampled  n. 4

Axis speed  ...(X/Y/Z/W/V) : X
```

In this case, track of axis X speed may be substituted by a different signal.

Hereafter, each data and the corresponding box are described.

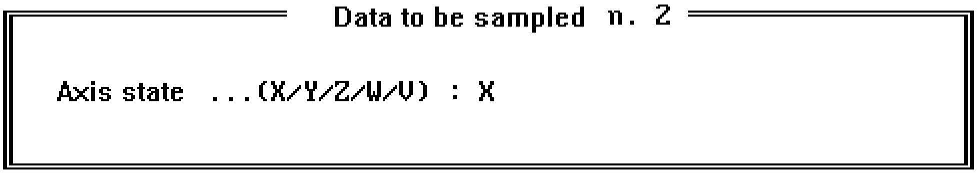
bitport: stores the state of the selected bitport



Data to be sampled n. 1

```
bitport  ...(0÷7+000÷255) : PENNA
```

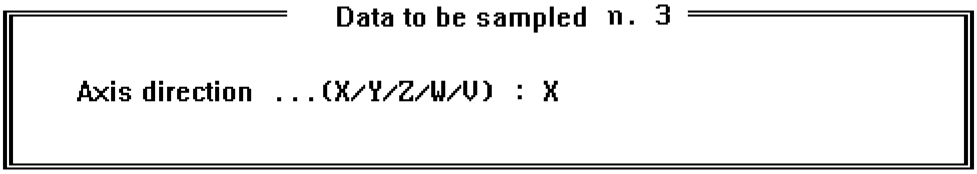
axis state: stores state of the selected axis; states are the same as provided for triggers



Data to be sampled n. 2

Axis state ...(X/Y/Z/W/V) : X

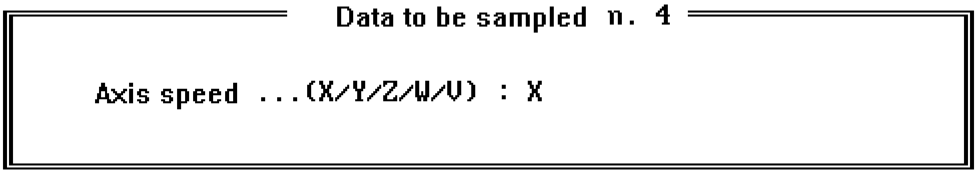
axis direction: stores direction of the selected axis.



Data to be sampled n. 3

Axis direction ...(X/Y/Z/W/V) : X

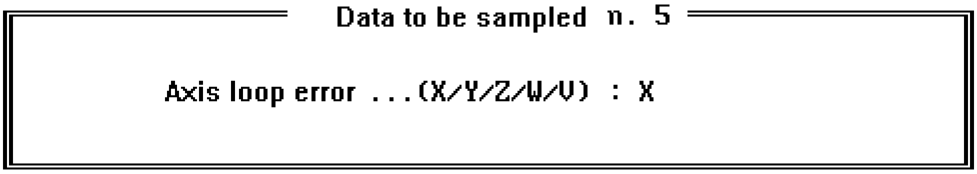
axis speed: stores speed of the selected axis.



Data to be sampled n. 4

Axis speed ...(X/Y/Z/W/V) : X

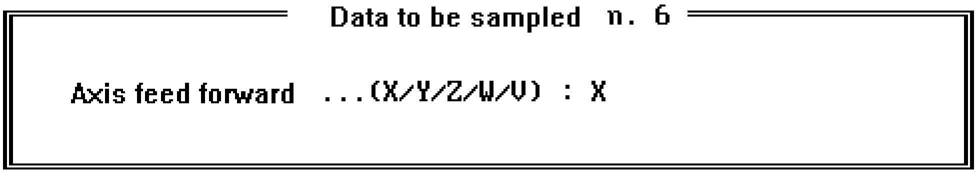
axis loop error: stores difference between theoretic and real coordinate of the selected axis.



Data to be sampled n. 5

Axis loop error ...(X/Y/Z/W/V) : X

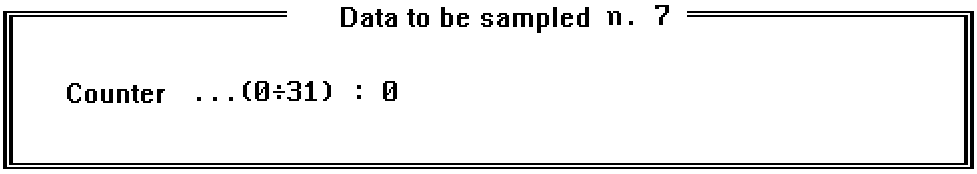
axis feed forward: stores feed forward applied to the selected axis.



Data to be sampled n. 6

Axis feed forward ...(X/Y/Z/W/V) : X

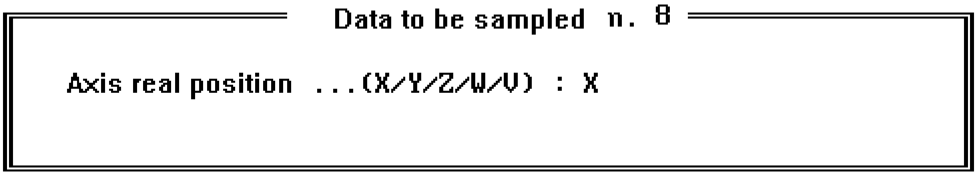
counter: stores value of the selected counter.



Data to be sampled n. 7

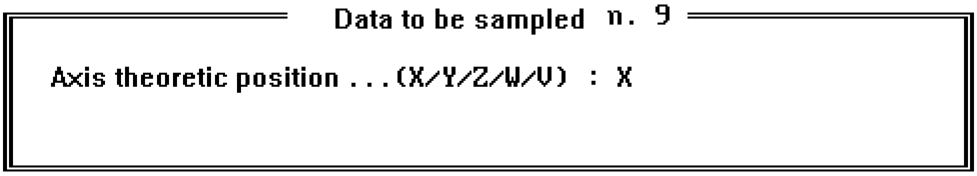
Counter ...(0÷31) : 0

axis real position: stores real position of the selected axis.



```
Data to be sampled n. 8

Axis real position ...(X/Y/Z/W/V) : X
```

axis theoretic position: stores theoretic position of the selected axis.



```
Data to be sampled n. 9

Axis theoretic position ...(X/Y/Z/W/V) : X
```

F6-VIEW

View operation allows displaying the sampled data in graphic form.

Description of screen display

Each signal on the screen is preceded by a description of the corresponding data sampled.

Cursor, constituted by a vertical bar, allows selecting a sample taken at a certain time delay with respect to the trigger event. Cursor may be displaced of one sample at a time with right/left arrow keys, or of 100 samples at a time with keys **CTRL+left arrow** or **CTRL+right arrow**.

Time is shown in the bottom frame on the left of the screen. An approximate indication can also be found on the time axis on the screen bottom.

Once a sampling is selected, the values of each selected signal may be read on the left side of the screen, under the description of the corresponding signal.

In the left area of the screen up to 12 frames are displayed, containing:

- in the upper zone, the description of the sampled signal

- in the lower zone, the value of the selected sample.

If the number of signals exceeds 12, the other signals may be seen with a vertical scroll of the screen, by means of keys **page up/down** or **arrow up/down**.

Keys available in view operation

ALT+H
Short description of the available keys.

ESC
Exits View operation.
If mouse is installed, the same effect may be obtained by clicking on the top left icon.

left arrow
Displaces the cursor bar of one sample to the left.
With mouse, click on the arrow on the left of the cursor bar.

right arrow
Displaces the cursor bar of one sample to the right.
With mouse, click on the arrow on the right of the cursor bar.

CTRL+ left arrow
Displaces the vertical bar of 100 samples to the left.
With mouse, click on the cursor bar.

CTRL+ right arrow
Displaces the vertical bar of 100 samples to the right.
With mouse, click on the cursor bar.

ALT+ left arrow
Horizontal scroll of a full screen to the left.
With mouse, click on the cursor bar.

ALT+ right arrow
Horizontal scroll of a full screen to the right.
With mouse, click on the cursor bar.

HOME
Sets the cursor bar on the first sample
With mouse, click on the cursor bar.

END
Sets the cursor bar on the last sample
With mouse, click on the cursor bar.

arrow up
Shifts up the highlighted signal.
With mouse, click on the description of the selected signal.

arrow down
Shifts down the highlighted signal.
With mouse, click on the description of the selected signal.

.

PAGE UP
Vertical scroll of the whole screen.

With mouse, click on the icon PgUp.

PAGE DOWN
Vertical scroll of the whole screen.
With mouse, click on the icon PgDn.

CTRL+ arrow down
Vertical scroll of the whole screen.
With mouse, click on the icon with arrow down.

CTRL+ arrow up Vertical scroll of the whole screen.
With mouse, click on the icon with arrow up.

CTRL+HOME Highlights first signal.
With mouse, click on the description of first signal.

CTRL+END Highlights last signal.
With mouse, click on the description of first signal.

TAB The highlighted signal is substituted by the following signal; this may be useful when not all the signals may be observed within the screen, as it allows displacing a signal from one page to another.
With mouse, click on the icon with the arrow down, on the right of the signal description.

SHIFT+TAB The highlighted signal is substituted by the preceding signal.
With mouse, click on the icon with the arrow up, on the right of the signal description.

F1 Zoom ON/OFF on the highlighted signal.
With mouse, click on the desired signal.

F2 Sets or resets interpolation of those signals that can only assume a discrete number of values (State of BitPort, Axis state, Axis direction).

F3 Sets or resets interpolation of analog signals (Axis speed, Counter Value, Axis coordinate).

F4 Restores the sequence of signals that have been displaced by means of keys TAB o SHIFT TAB.

Description of error messages.

During View operation, some error messages may appear. Each one is shortly described hereafter.

- **WRONG RELEASE OF LANGUAGE FILE**
  The version of the language file is not correct, or not updated.
  Load the correct language file.

- **Data file not accessible !!**
  The analyser did not find the file containing sampled data.
  Execute a new Sample and Store.

- **Data file not correctly stored !!**
  The analyser found the file containing sampled data, but these are not correctly recorded.
  Execute a new Sample and Store.

- **Incorrect number of signals !!**
  The analyser found the file containing sampled data, but these are not correctly recorded. Execute a new Sample and Store.

- **Incorrect number of intervals !!**
  The analyser found the file containing sampled data, but these are not correctly recorded.
  Execute a new Sample and Store.

- **Not enough Memory!!**
  Not enough memory for graphic display.
  If possible release some memory.

- **Error in color definition: message n. 1059 !!**
  Within language files, colors are indicated for some sections of the program; some of these colors are not correctly recorded. Substitution of language file is recommended.

- **Error in color definition: message n. 1061 !!**
  Within language files, colors are indicated for some sections of the program; some of these colors are not correctly recorded. Substitution of language file is recommended.

USE OF ANALYSER

Execution of a track requires the following procedure:

- Access to Analyser from **Automatic**, pressing keys **ALT+O**; when accessing from **Programming** operation, press keys **CTRL+ALT+O**.

- Digitize the name of the file where sampled data should be stored.

- Introduce basic parameters, triggers and data to be sampled; all data will be displayed on the screen (see following page).

- Start sampling pressing function key F1-START.
  On the top right corner the message "**sampling in progress**" will be displayed, informing that storing is not yet complete or triggers have not yet been reached.

- If program is not under execution, start it coming back to Automatic operation and pressing function key F1-START or "field start".

- If program is under execution, You may exit Analyser operation without losing the current track, and enter Manual or Console operation and work normally on the machine.

- Coming back to Analyser, if message "sampling in course" has disappeared, You may store sampled data by pressing function key F3-STORE; this operation requires some time, that depends on the number of samples. All data will be stored in the file named as previously specified.

- Once storage has been completed, data may be displayed by pressing function key F6- VIEW. Data will be displayed in the graphic form shown in next page.

APPENDIX B

Analyser

GENERAL DESCRIPTION

The analyser functionality allows storing and displaying in graphic form the time behavior of physical entities controlled by the system. This functionality may be accessed in **Automatic** mode only, while within Program Management functionality the recorded samples may be displayed:

The analyser samples one or more variables at selected time intervals; it works off-line, so it doesn't allow displaying the sampled signals in real time, but it stores data that can be analyzed later.

Analyser operation is constituted by two main blocks: definition of data to be sampled and stored, and graphic display of sampled data.

Data storage uses memory area named "others", so that sufficient space in this area must be provided [1].

The more space is configured in this area, the higher will be the number of data sampled, as the analyser will go on storing data until the assigned memory area is full.

In order to start a track, the operator must define:

- basic parameters
- triggers
- data to be sampled.

Access to these data is provided by function key F5-EDIT.

Data display is provided by function key F6-VIEW.

Data **Bitport** and **Counter Number** may also be set by using the corresponding symbolic definition.

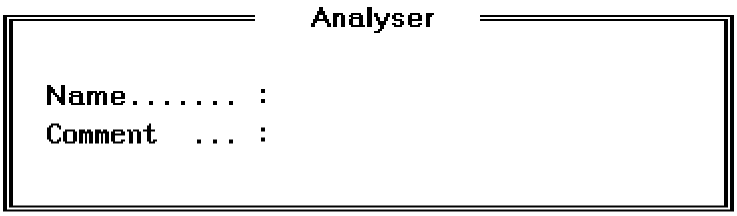
[1]. See **Memory areas used on board** in Station Configuration.

ENTERING ANALYSER OPERATION

Analyser functionality is activated in **Automatic,** by pushing keys **ALT+O**, even while automatic program is running.

The analyser works on the active station, but station can be changed by pressing function key **F9-CHANGE**.

The following box will be displayed:

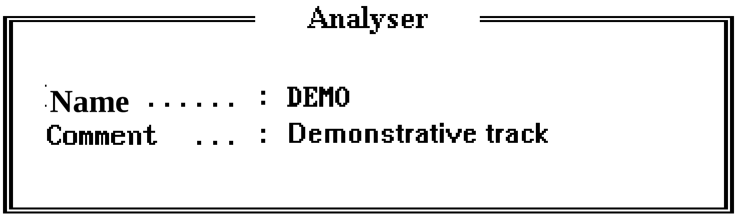


Analyser

Name....... :
Comment ... :

Creation of a new track

For creation of a new track, digitize filename and, if desired, a comment, as indicated in the box below:



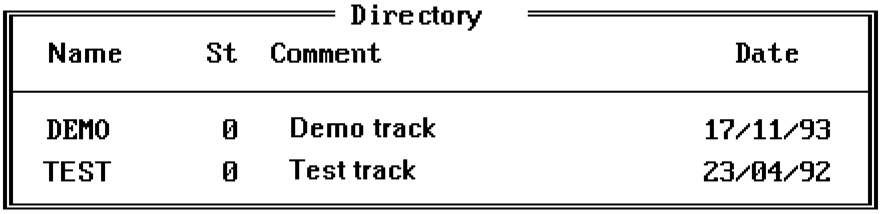
Analyser

Name ...... : DEMO
Comment ... : Demonstrative track

In this case, the system will create a new file containing both sampled data and parameters corresponding to the track.. Data will be stored after sampling pressing function key **F3-STORE**.

Selection of an existing track

For working on an existing track, press key **ENTER** in the first box; the screen will display a list of the existing files in the active station:



Directory

Name	St	Comment	Date
DEMO	0	Demo track	17/11/93
TEST	0	Test track	23/04/92

Select a file and press **ENTER**. All data used for that track will appear on the screen; You can modify them by pressing function key **F5-EDIT** in order to access to Edit menu.

DESCRIPTION OF FUNCTION KEYS

Function Keys available are the following:

F1 - START Verifies trigger and starts sampling; on the top right corner the screen will display the message: "sampling in progress".

F2 - END Exits trigger verification or data sampling before completing track; all data sampled will be lost

F3 - STORE Stores track in a file

F4 - FILE Allows editing a new file or displaying an existing file. The procedure is the same as described for entering analyser operation.

F5 - EDIT Allows editing parameters.

F6 - VIEW Displays track data in graphic form.

F9 - CHANGE Allows sampling in a different station.

F10 - EXIT Exits analyser operation.

Almost all functions associated to function keys are self explaining; a more detailed explanation is given hereafter for **F6-EDIT** and **F7-VIEW**.

F5-EDIT

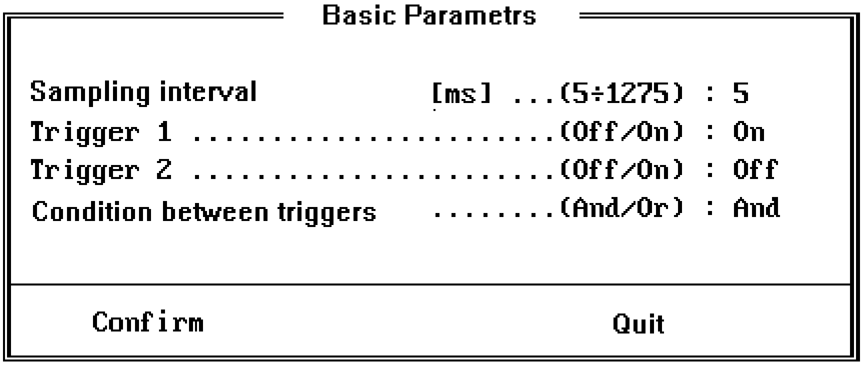
Edit operation allows introducing or modifying **basic parameters**, **triggers** and **data to be sampled**. The menu is the following:

Errore. ncorporato non è valido.

Basic Parameters

Basic Parameters allow configuring the analyser. Specifically, they define sampling interval, triggers (one or two) and logical correlation between triggers: **OR** if sampling should start as soon as one of the two triggers is verified, **AND** if both of them should be verified.

The following box will be displayed:



Basic Parametrs

```
Sampling interval          [ms] ...(5÷1275) : 5
Trigger 1 ......................(Off/On) : On
Trigger 2 ......................(Off/On) : Off
Condition between triggers  ........(And/Or) : And
```

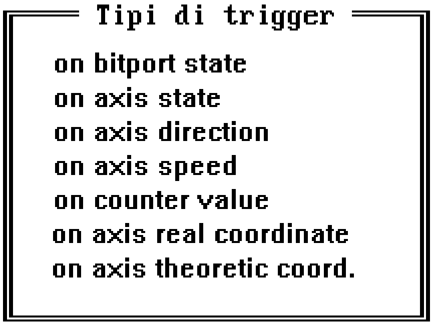
Sampling Interval: Time in milliseconds between each sampling; minimum time corresponds to real time period[2].

Trigger 1 and 2: Defines trigger state: On or Off. When Off, trigger is not considered.

Condition between triggers: Defines the logical condition between triggers.

Trigger 1 and 2

Triggers are used for starting storage; various types of triggers may be selected within the following menu:

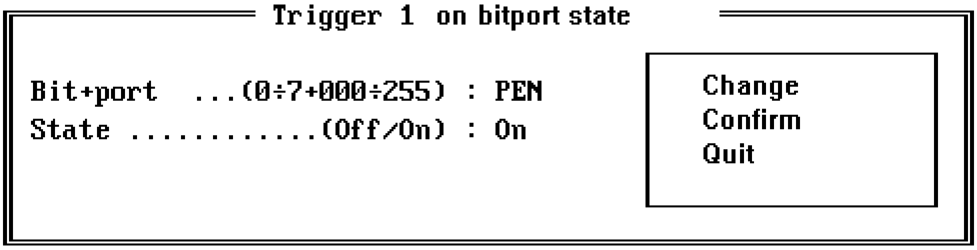


Each item and the corresponding box are described hereafter.

[2]. See **Working frequencyes** in General module Configuration.

1) on bitport state: storage begins when the selected bitport is in the state defined;

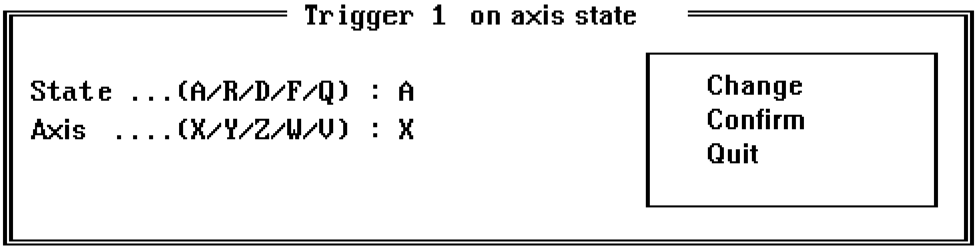
inputs, outputs and flags may be selected.



```
Trigger 1  on bitport state

Bit+port  ...(0÷7+000÷255) : PEN
State ............(Off/On) : On
```

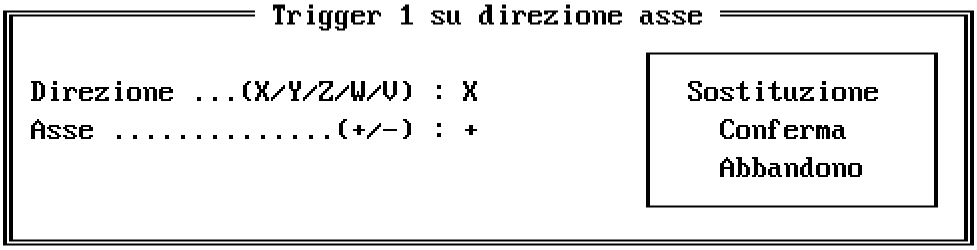
2) on axis state: storage starts when axis is in the selected state, axes may be tested in the following states:

A	Acceleration
R	Regime
D	Deceleration
F	Theoretic movement completed
Q	Axis in position



```
Trigger 1  on axis state

State ...(A/R/D/F/Q) : A
Axis  ....(X/Y/Z/W/V) : X
```

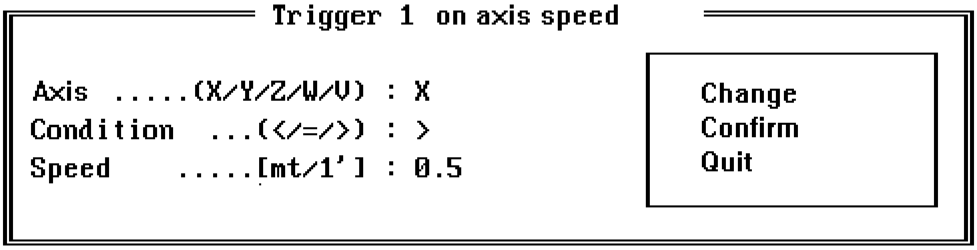
3) on axis direction: storage starts when axis is moving or beginning to move in the selected direction; axes can be tested in positive (+) and negative (-) direction.



```
Trigger 1 su direzione asse

Direzione ...(X/Y/Z/W/V) : X
Asse .............(+/-) : +
```

4) on axis speed: storage begins when axis runs at a speed corresponding to the condition defined, with reference to programmed speed; axes may be tested with respect to the following conditions:

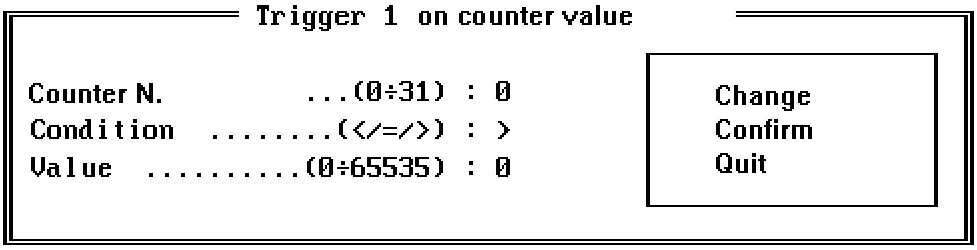
<	axis speed lower than indicated
=	axis speed equal to indicated
>	axis speed greater than indicated



```
Trigger 1 on axis speed

Axis .....(X/Y/Z/W/V) : X
Condition ...(</=/>) : >
Speed .....[mt/1'] : 0.5
```

5) on counter value: storage begins when the selected counter contains a value that satisfies the condition indicated, with respect to the programmed number:

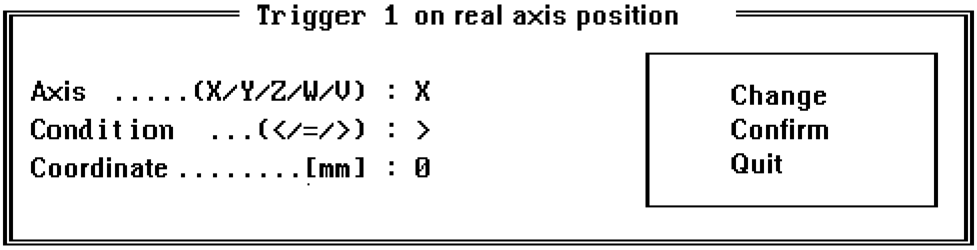
<	counter lower than programmed number
=	counter equal to programmed number
>	counter greater than programmed



```
Trigger 1 on counter value

Counter N.          ...(0÷31) : 0
Condition  ........(</=/>) : >
Value  ..........(0÷65535) : 0
```

6) on real axis coordinate: storage begins when real position of selected axis satisfies the condition indicated:

<	axis coordinate lower than programmed
=	axis coordinate equal to programmed
>	axis coordinate greater than programmed

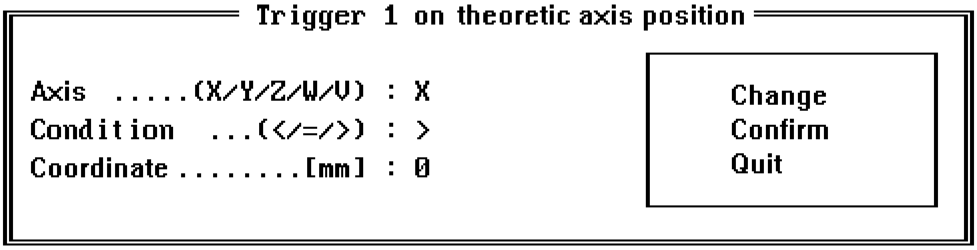


Trigger 1 on real axis position

```
Axis  .....(X/Y/Z/W/V) : X
Condition  ...(</=/>) : >
Coordinate ........[mm] : 0
```

7) on theoretic axis position: storage begins when theoretic position of selected axis satisfies the condition indicated:

<	axis coordinate lower than programmed
=	axis coordinate equal to programmed
>	axis coordinate greater than programmed



Trigger 1 on theoretic axis position

```
Axis  .....(X/Y/Z/W/V) : X
Condition  ...(</=/>)  : >
Coordinate ........[mm] : 0
```

Change of a trigger type

For changing a type of trigger, select **Change** within the box; the menu of available types of trigger will be displayed.

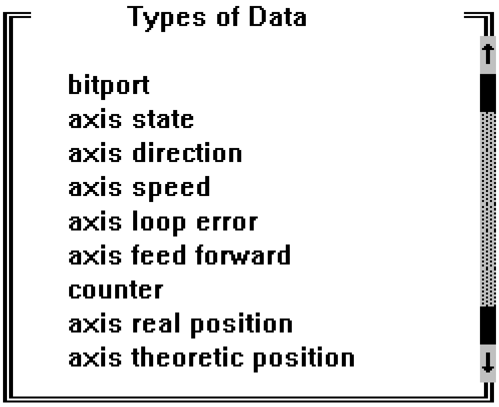
Data to be sampled

In this section one may define all data that shall be stored after trigger; the more data one selects, the lower will be the number of samples that may be stored.

Each data may occupy 1 or 2 or 4 bytes in memory; specifically:

- 1 byte for: bitport, axis state, axis direction.
- 2 bytes for: axis speed, axis loop error, axis feed forward, counter.
- 4 bytes for: axis real position, axis theoretic position

The following types of data are available:



Types of Data

bitport
axis state
axis direction
axis speed
axis loop error
axis feed forward
counter
axis real position
axis theoretic position

A maximum of 16 data may be sampled. Within box, data is selected by means of **cursor keys**, **TAB** key allows selecting the button for the function desired, that is activated by **ENTER**.

The buttons provide the following functions:

Change Allows substituting current data with a new one.
Under the box, the menu of Types of Data is displayed, for selection of the new type.

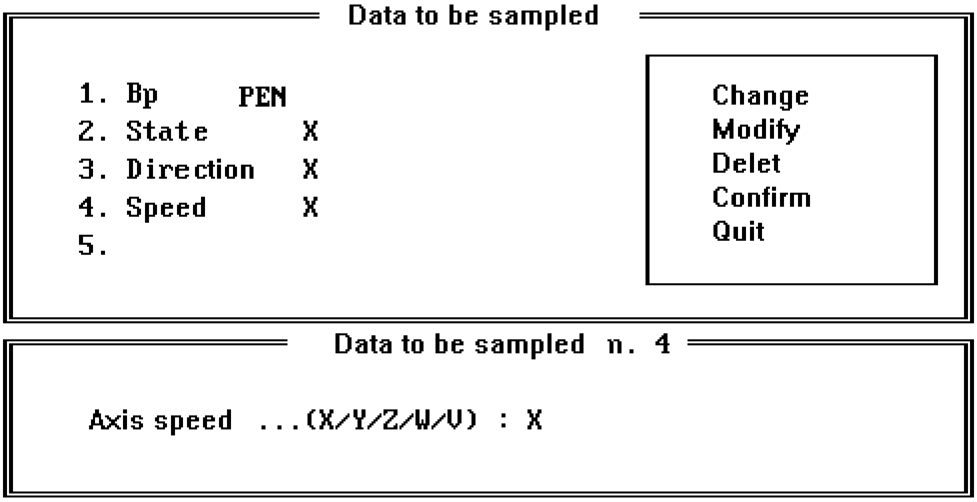
Modify Allows modifying current data with another data.
The box containing the preceding data is displayed, so that it can be modified.

Delete Deletes the selected data.

Confirm Confirm modifications and terminates editing of Data to be sampled

Quit Exits without saving modifications.

The following boxes show how data number 4 may be **Modified**:

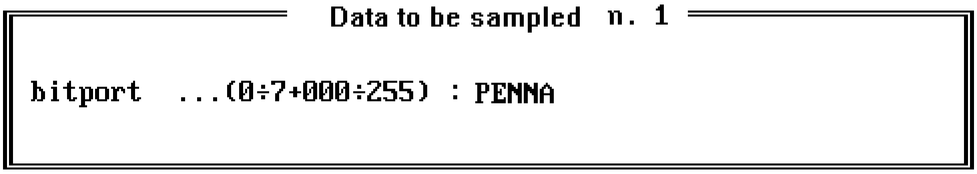


```
                    Data to be sampled

   1. Bp      PEN                          Change
   2. State       X                        Modify
   3. Direction   X                        Delet
   4. Speed       X                        Confirm
   5.                                      Quit

                  Data to be sampled  n. 4

   Axis speed  ...(X/Y/Z/W/V) : X
```

In this case, track of axis X speed may be substituted by a different signal.

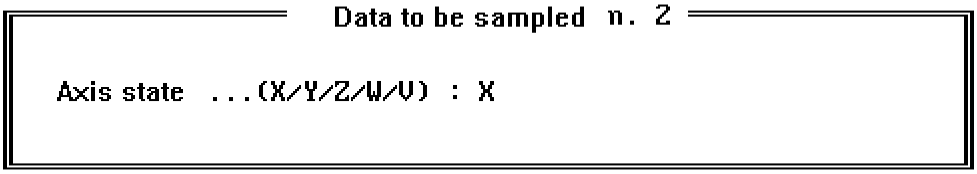
Hereafter, each data and the corresponding box are described.

bitport: stores the state of the selected bitport



```
Data to be sampled  n. 1

bitport  ...(0÷7+000÷255) : PENNA
```

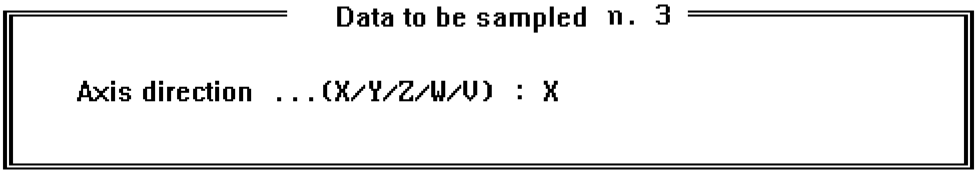
axis state: stores state of the selected axis; states are the same as provided for triggers



Data to be sampled n. 2

Axis state ...(X/Y/Z/W/V) : X

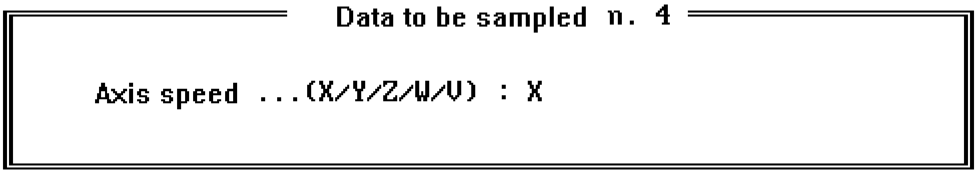
axis direction: stores direction of the selected axis.



Data to be sampled n. 3

Axis direction ...(X/Y/Z/W/V) : X

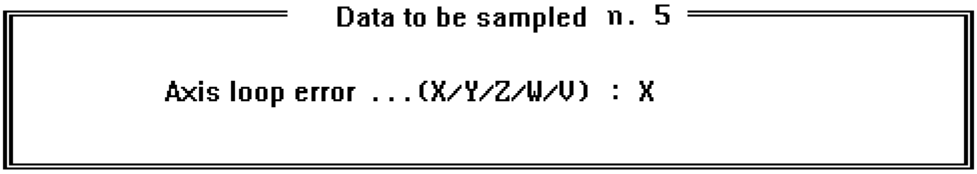
axis speed: stores speed of the selected axis.



Data to be sampled n. 4

Axis speed ...(X/Y/Z/W/V) : X

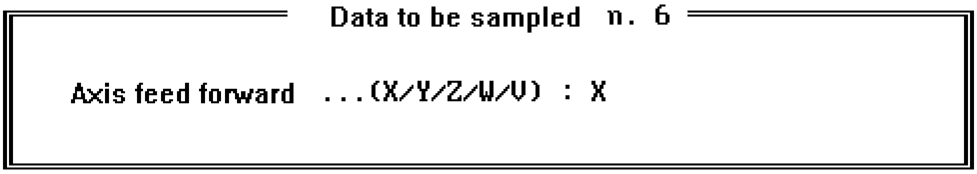
axis loop error: stores difference between theoretic and real coordinate of the selected axis.



Data to be sampled n. 5

Axis loop error ...(X/Y/Z/W/V) : X

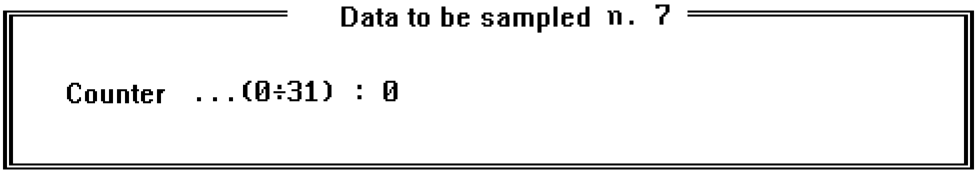
axis feed forward: stores feed forward applied to the selected axis.



Data to be sampled n. 6

Axis feed forward ...(X/Y/Z/W/V) : X

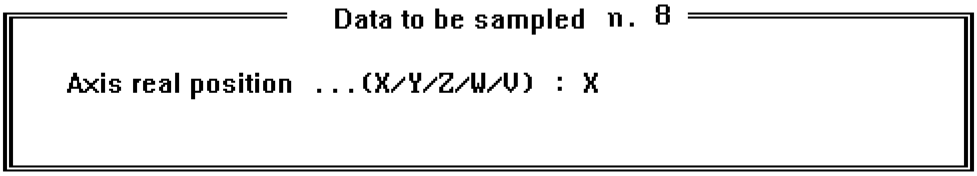
counter: stores value of the selected counter.



Data to be sampled n. 7

Counter ...(0÷31) : 0

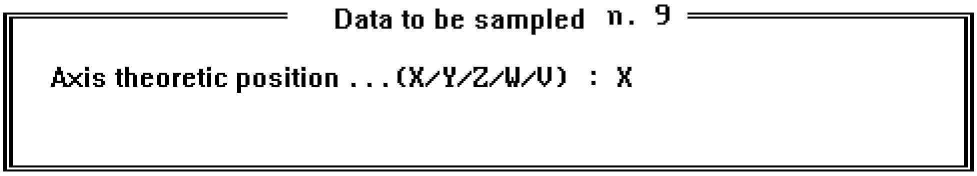
axis real position: stores real position of the selected axis.



Data to be sampled n. 8

Axis real position ...(X/Y/Z/W/V) : X

axis theoretic position: stores theoretic position of the selected axis.



Data to be sampled n. 9

Axis theoretic position ...(X/Y/Z/W/V) : X

F6-VIEW

View operation allows displaying the sampled data in graphic form.

Description of screen display

Each signal on the screen is preceded by a description of the corresponding data sampled.

Cursor, constituted by a vertical bar, allows selecting a sample taken at a certain time delay with respect to the trigger event. Cursor may be displaced of one sample at a time with right/left arrow keys, or of 100 samples at a time with keys **CTRL+left arrow** or **CTRL+right arrow**.

Time is shown in the bottom frame on the left of the screen. An approximate indication can also be found on the time axis on the screen bottom.

Once a sampling is selected, the values of each selected signal may be read on the left side of the screen, under the description of the corresponding signal.

In the left area of the screen up to 12 frames are displayed, containing:

- in the upper zone, the description of the sampled signal

- in the lower zone, the value of the selected sample.

If the number of signals exceeds 12, the other signals may be seen with a vertical scroll of the screen, by means of keys **page up/down** or **arrow up/down**.

Keys available in view operation

ALT+H Short description of the available keys.

ESC Exits View operation.
If mouse is installed, the same effect may be obtained by clicking on the top left icon.

left arrow Displaces the cursor bar of one sample to the left.
With mouse, click on the arrow on the left of the cursor bar.

right arrow Displaces the cursor bar of one sample to the right.
With mouse, click on the arrow on the right of the cursor bar.

CTRL+ left arrow Displaces the vertical bar of 100 samples to the left.
With mouse, click on the cursor bar.

CTRL+ right arrow Displaces the vertical bar of 100 samples to the right.
With mouse, click on the cursor bar.

ALT+ left arrow Horizontal scroll of a full screen to the left.
With mouse, click on the cursor bar.

ALT+ right arrow Horizontal scroll of a full screen to the right.
With mouse, click on the cursor bar.

HOME Sets the cursor bar on the first sample
With mouse, click on the cursor bar.

END Sets the cursor bar on the last sample
With mouse, click on the cursor bar.

arrow up Shifts up the highlighted signal.
With mouse, click on the description of the selected signal.

arrow down Shifts down the highlighted signal.
With mouse, click on the description of the selected signal.
.

PAGE UP Vertical scroll of the whole screen.

	With mouse, click on the icon PgUp.
PAGE DOWN	Vertical scroll of the whole screen. With mouse, click on the icon PgDn.
CTRL+ arrow down	Vertical scroll of the whole screen. With mouse, click on the icon with arrow down.

CTRL+ arrow up
Vertical scroll of the whole screen.
With mouse, click on the icon with arrow up.

CTRL+HOME
Highlights first signal.
With mouse, click on the description of first signal.

CTRL+END
Highlights last signal.
With mouse, click on the description of first signal.

TAB
The highlighted signal is substituted by the following signal; this may be useful when not all the signals may be observed within the screen, as it allows displacing a signal from one page to another.
With mouse, click on the icon with the arrow down, on the right of the signal description.

SHIFT+TAB
The highlighted signal is substituted by the preceding signal.
With mouse, click on the icon with the arrow up, on the right of the signal description.

F1
Zoom ON/OFF on the highlighted signal.
With mouse, click on the desired signal.

F2
Sets or resets interpolation of those signals that can only assume a discrete number of values (State of BitPort, Axis state, Axis direction).

F3
Sets or resets interpolation of analog signals (Axis speed, Counter Value, Axis coordinate).

F4
Restores the sequence of signals that have been displaced by means of keys TAB o SHIFT TAB.

Description of error messages.

During View operation, some error messages may appear. Each one is shortly described hereafter.

- **WRONG RELEASE OF LANGUAGE FILE**
  The version of the language file is not correct, or not updated.
  Load the correct language file.

- **Data file not accessible !!**
  The analyser did not find the file containing sampled data.
  Execute a new Sample and Store.

- **Data file not correctly stored !!**
  The analyser found the file containing sampled data, but these are not correctly recorded.
  Execute a new Sample and Store.

- **Incorrect number of signals !!**
  The analyser found the file containing sampled data, but these are not correctly recorded. Execute a new Sample and Store.

- **Incorrect number of intervals !!**
  The analyser found the file containing sampled data, but these are not correctly recorded.
  Execute a new Sample and Store.

- **Not enough Memory!!**
  Not enough memory for graphic display.
  If possible release some memory.

- **Error in color definition: message n. 1059 !!**
  Within language files, colors are indicated for some sections of the program; some of these colors are not correctly recorded. Substitution of language file is recommended.

- **Error in color definition: message n. 1061 !!**
  Within language files, colors are indicated for some sections of the program; some of these colors are not correctly recorded. Substitution of language file is recommended.

USE OF ANALYSER

Execution of a track requires the following procedure:

- Access to Analyser from **Automatic**, pressing keys **ALT+O**; when accessing from **Programming** operation, press keys **CTRL+ALT+O**.

- Digitize the name of the file where sampled data should be stored.

- Introduce basic parameters, triggers and data to be sampled; all data will be displayed on the screen (see following page).

- Start sampling pressing function key F1-START.
  On the top right corner the message "**sampling in progress**" will be displayed, informing that storing is not yet complete or triggers have not yet been reached.

- If program is not under execution, start it coming back to Automatic operation and pressing function key F1-START or "field start".

- If program is under execution, You may exit Analyser operation without losing the current track, and enter Manual or Console operation and work normally on the machine.

- Coming back to Analyser, if message "sampling in course" has disappeared, You may store sampled data by pressing function key F3-STORE; this operation requires some time, that depends on the number of samples. All data will be stored in the file named as previously specified.

- Once storage has been completed, data may be displayed by pressing function key F6- VIEW. Data will be displayed in the graphic form shown in next page.

UPDATING TO 4.01 and 4.2 VERSIONS

If the present SW version is installed as updating a 4.01 or 4.2 versions, or it uses the working disks of these versions, after installing, the following operations must be performed:

1) enter in the **Station setup**, for any station, and select **Board used memory areas**. Modify the various areas data as the total was equal or less than **27648** bytes.

2) the standard interpolation Frequency and the Axes Interpolation control loop are changed from 500Hz to 400Hz, then, entering in the **General setup**, for any module, and select **Working frequencies**. Insert in the **Interpolation frequency** and in the **Axes interpolation control loop** the new value **400** and exit.

3) run Compilation of all the functions and part programs, included in the working disk

4) if the boards are equipped with bactery backup RAM, at the first initializing, select the Parameters Re-transmission by the F4 - RETRY key.

UPDATING TO 4.01 VERSIONS

If the present SW version is installed for updating old versions, before the 4.01 dated 5/7/91, or uses the previous working disks, the following setup modifyng must be performed, after installing:

if I/O expansion cards are equipped, enter in the **Station setup**, for any station, and select **Input/output expansion,** modify the first card typology (i.e. from INOUTR to IOMOD) and exit to force automatically the Parameters Compilation.
Then return and restore the original card tipology and exit, to recall the Parameters Compilation with the correct card outfit.

UPDATING TO 4.2 VERSIONS

If the present SW version is installed for updating 4.2, 4.2a or 4.2b versions, or uses the previous working disks, the following setup modifyng must be performed, after installing:

1) enter in the **Module Setup**, for any module, and select **Module Options**. At the item **Real quotes displaying** select **No**.

2) enter in the **Station setup**, for any station, and select **Feedrate override**. Insert a new any value, exit and re-entry restoring the original value.

UPDATING TO 4.3 and previous VERSIONS

If the present SW version is installed for updating 4.3, 4.2a or 4.2b, 4.01 versions, or uses the previous working disks, the following setup modifyng must be performed, after installing:

1) enter in the **System Setup** and modify the Baud-rate value to 19200.

2) enter in the **General setup ,** for any module, and insert the parameter Bactery = No,
   recall the parameters transmission by the F4 - RETRY key, in the System Initializing, then return in the General setup and restore Bactery = Yes.

THE GPL1000 LANGUAGE

THE LANGUAGE

The GPL1000 (General Purpose Language) is the programming language for the PTP1000 system.

The GPL1000 includes an instructions set may be used to define the executable programs for PTP200N CPU card and, with some limitations, for the PLC200 card. In the following list, only the **p** character marked instructions are available for the PLC200 card programs.

• Auxiliary Instructions

DELAY	p	Programmable delay
END	p	End of program
ERROR	p	Cycle error
MESSAGE	p	Indexed message display
NOP	p	No operation
PAUSE	p	Program pause
RESTART	p	Special firmware subroutine call
SPECIAL	p	Special instruction
VIDEO	p	Direct message display

• Counter/Timer instructions

DCOUNT	p	Counter decrement
HTIMER	p	Stop counter
ICOUNT	p	Counter increment
INCOUNT	p	Copy a counter into an OUT port/ports
OUTCOUNT	p	Copy into a counter from an IN port/ports
RTIMER	p	Timer reset
SCOUNT	p	Set counter
STIMER	p	Start timer
TCOUNT	p	Test counter
TTIMER	p	Test timer
VCOUNT	p	Counters display

• Branch instructions

BRA	p	Branch to Label
CALL	p	Subroutine call
ENDREP	p	End of Repeat bloc
FCALL	p	Function Call
FRET	p	Return from Function
REPEAT	p	Begin Repeat bloc
RET	p	Return from Subroutine

• Multiprogramming Instructions

APROG	p	Parallel tasks starting
EPROG	p	End of parallel task
FPROG	p	Parallel functions starting
RPROG	p	Parallel task/function restarting
SPROG	p	Parallel task/function suspending
TPROG	p	Parallel task/functions execution status test
WPROG	p	Waiting for parallel task/function end

• Input/Output control instructions

AND	p	Logical AND
COMP	p	Port COMPARE
INP	p	Input port read
NOT	p	Logical NOT
OR	p	Logical OR
OUT	p	Output port write
RES	p	Bit of Port Reset
SET	p	Bit of Port Set
SHIFTL	p	Shift Left of a port (Rotate)
SHIFTR	p	Shift Right of a port (Rotate)
SKZ	p	Skip if (bit/port) is 0
SNZ	p	Skip if (bit/port) is not 0
SYNC	p	Synchronisme Send
VOUT		Analog output write
WIZ	p	Wait if (bit/port) is 0
WNZ	p	Wait if (bit/port) is not 0
WSYNC	p	Synchronisme Wait
XOR	p	Logical Exclusive-OR

• Axes moving Instructions

ABS	p	Absolute axes moving
ACCEL		Acceleration set
ADDOFS		Axes Offsets Addition
ADJUST		Axis position Adjust
CALOFS		Axis Offset autolearning
CHAIN		Axes Chain
FREE		Axes Frees (Open loop)
GAIN		Axis Gain
INC	p	Incremental axes moving
LIMITOFF		Axes Limits control Disable
LIMITON		Axes Limits control Enable
MOV	p	Axes Move to programmed quotes
MOVOFS		Axes move to Offset quotes
NORMAL		Axes in normal Control Loop
SERVO		Servo Error control Enable (for point to point moving)
SETFF		Feed forward set
SETQ	p	Axis quote Set
SETOFS		Axis Offset Set
SETPF		Flying Setpoint
SKE	p	Axis / Interpolation Status Test
STOP	p	Axes Stop
SUBOFS		Axes Offsets Subtract
TOFS		Axis Offset Test
TQ	p	Axis quote Test
USEOFS		Axes Offsets recalling
VEL		Axis Speed set
WEND	p	Wait for Axis/Interpolation status

• Interpolation instructions

CIRCLE		Two axes Circular Interpolation
ECNT		End of Counturing
HELIC		Three axes Helycoidale Interpolation
ICNT		Start of Counturing
LINEAR		Two/Three axes Linear Interpolation

• Tables of Quotes instructions

DPTAB		Table Pointer decrement
EQTAB		Move to pointed table location quote
IPTAB		Table Pointer increment
MQOFS		Copy table quote into Offset
MQTAB		Copy a quote into a Table
SPTAB		Table Pointer set
TPTAB		Table Pointer test
TQTAB		Compare axe quote with table quote

THE INSTRUCTIONS SET

In the following sections alls the different GPL1000 instructions are described.

For everyone are defined:

- the Name
- the Title
- the Syntax format
- the Arguments description
- the Operating description
- some using exemples

In the description, the following symbols are used:

[,] optional argument/s.

x.x numerical value with decimals.

axis DC motor axis name
X Y Z W V.

ACaxe AC motor axes name:
X1 Y1 Z1 W1 V1 U1 X2÷U2 X3÷U3;
(in the case of X1÷U1 axes, the number may be omitted, since automatically assumed by the Compiler).

bitport bit+port (bit=0÷7, port=000÷255) address.

dataport data+port (data=000÷255, port=000÷255) value.

label Instruction or programs identifier used for Branch and Call instructions: (maximum length of 15 characters including all alphanumeric **_ - < >** symbols).
When used to identify an istruction, the label must be followed by the '**:**' character; the following field may include an instruction or not. This last case is suggested when the Autolearning operation mode is used.

If the instruction requires more arguments, these must be separated by the “,” (comma) or “ “ (space) character (i.and.: WEND X,Q).

ABS

Selects the Absolute quote programming mode, for the axes moving.

ABS axes

ABS ACaxe

axes: **X Y Z W V**	Names of the selected axes.
ACaxe: **X1÷U3**	Axis name.

After this instruction, alls the following point to point programmed movements of the **axes** are assumed in Absolute mode.

See also the instruction: INC.

ACCEL

Defines the axis acceleration value.

ACCEL axis[,value]

ACCEL I[,value]

axis: **X Y Z W V**	**A**xis name.
interpolation: **I**	Interpolation direction.
value: **x**	Acceleration time (in millisec.)

After this instruction is executed, the following point to point **axis** movements assume the programmed acceleration time.

The new acceleration value is accepted only in the stopped axe condition, otherwise the end of the movement is waited.

The programmed acceleration value must be equal or larger of the default value stated in the Station Setup mode: this last value is automatically assumed when programming is missing.

The acceleration value defines the time needed to rise from the 0 speed to the maximum speed value stated in the Station Setup mode.

The same considerations are valid to programming the **interpolation** acceleration.

ADJUST

Servodrive axis offset making up.

ADJUST axes

axes: **X Y Z W V**	Selected axes names.

Allows to detect and correct the analog offset present on the reference voltage required by the servodrive of the selected axes.

This instruction must be recalled in axis quiescent condition (“Q” status): on the contrary the end of the movements is waited. The measure time is approximatively 200 milliseconds.

The offset compensation value is stored and may be texted once by successive ADJUST instructions.

ADDOFS

Two programmable axes offsets add

ADDOFS axis1,offset1,axis2,offset2[,axis3,offset3]

axis1: **X Y Z W V**	Source axis 1 name.
offset1: **0÷31**	Source offset 1 number.
axis2: **X Y Z W V**	Source axis 2 name.
offset2: **0÷31**	Source offset 2 number.
axis3: **X Y Z W V**	Destination axis name.
offset3: **0÷31**	Destination offset number.

This instruction may be used to add two quotes stored in the axis offset registers.

If the destination offset register number is missing, the result will be stored in the offset 2 register.

This operation may also include offset register related to differents axes: in this case the axes must have the same encoder resolution.

See also the instructions: SETOFS, USEOFS, CALOFS, SUBOFS, TOFS, MOVOFS.

AND

2 bits Logical AND

AND op1,op2[,result]

op1: **bitport**	Operand 1.
op2: **bitport**	Operand 2.
result: **bitport**	Result

Makes the logical AND enter the operands 1 and 2 (**op1** and **op2**) storing the result in the second operand or in the **result** bit, if programmed.

APROG

Runs one or more parallel tasks.

APROG num

APROG num,program names

APROG program names

num: **1÷8**	Parallel tasks identification numbers.
program names:	Parallel tasks names.

This instruction, as the FPROG, EPROG, SPROG, RPROG, TPROG and WPROG, is used in the multitasking mode.

Allows to run more programs (max 4) or tasks in contemporary.

If, in the instruction, the parallel programs numbers (**num**) are indicated, the eventual names are automaticaly associated to the available tasks. Obviously, if more programs are recalled for execution, the instruction waits until all the required tasks are available to begin the operation, in manner to obtain a contemporary start of all the programs.

Each program is separated by the others using the **PROG num** or **PROG programname** and ended by the EPROG instruction.
The Main Program (1) must be ended by the END instruction.

When only one program is actived (then the multitasking mode not used), as a matter of fact the task 1 is used: then the system considers that the PROG instruction, heading the program, is equivalent to PROG 1, assumed by default if missing. Indeed, in the multitasking case, the first program (PROG or PROG 1) is assumed as main, from which alls the others are started, and automaticaly activec by the system.

If the start of a program already in execution is attempted, the instruction waits the end before to restart, then next instruction is scheduled.

See also the instructions: EPROG, FPROG, RPROG, SPROG, TPROG, WPROG.

BRA

Jump to a selected label.

BRA label

label: Name of the label of the instruction.

The program execution restarts from the **labelled** instruction.

The label (maximum 15 alphanumeric characters) must be followed by the “:” character and mays be placed:

- on a avoid line, *before* the corresponding instruction line (compulsory, if the AUTOLEARNING operation mode is used)

  i.e.:
  ```
  LABEL:
              X      10,Q
  ```

- or at the beginning of the instruction line

  i.e.:
  ```
  LABEL:      X      10,Q
  ```

It’s possible to make different branchs to the same label, but it is forbidden to insert, in the same program, two labels with the same name: in this case, the Compiler will generate the DOUBLE LABEL DEFINITION error message.

If some “Branch” instruction refers to an inexistent label, the LABEL NOT FOUND error message will be displayed by the Compiler.

For instance:

```
            PROG
            SKZ         FCPST
            BRA         EXEMPLE1
            SET         PST
            BRA         FINE
EXEMPLE1:   RES         PST
FINE:
            END
```

CALL

Subroutine call

CALL subroutine

subroutine: Name of the recalled subroutine.

The program execution restarts from the first recalled **subroutine** statement

The subroutine is named by the Label placed on it first instruction. Each subroutine must end with the RET instruction, that allows to return to a calling program, restarting execution from the statement following the CALL instruction.

Four levels of subroutine nesting is allowed.

See also the instruction: RET.

For instance:

X axis Zero point search.

```
            PROG
            ABS         X
            VEL         X,0.5
            CALL        SETPX
            END

;*** subroutine SETPX ***

SETPX:      X           100
            WIZ         TC0X
            SETQ        X,0
            RET
```

CALOFS

Offset quote (for correction) Autolearning

CALOFS axis,offset,sign,bitport,status,quote

axis: **X Y Z W V**	Axis name.
offset: **0÷31**	Destination Offset register number.
sign: **+ -**	Defines the rule for the correction offset storing.
bit: **0÷7**	Bit number and..
port: **000÷255**	Port address of the signal line.
status: **0 1**	Selects: 0 bitport active if closed 1 bitport active if open
quote: **±x.x**	Nominal quote.

This instruction allows autolearning of a correction offset computed as difference enter the programmed **Nominal quote** and the **effective absolute quote** (of the selected axis), sampled in correspondence of the signal switching defined by the **status** field (from disactive to active status) and the **bitport** address (of the triggering signal),then:

correction offset = effective quote - nominal quote

The resulting value is stored in the selected **offset** register, according to the programmed **sign** value, and precisely:

positive sign (+): the correction offset is stored with the computed sign
negative sign (-): the correction offset is stored with exchanged sign.

This computed correction offset is also transmitted to all the other cards of the system and stored in the corresponding axis offset register, as programmed. In this mode, if alls the axes encoders resolutions are the same, this instruction allows autolearning this correction factor by a single axis movement automatically updating all the corresponding axes on the different cards.

See also the instructions: SETOFS and USEOFS.

CHAIN

Two (or more) axes movements chain.

CHAIN axis,axes

axis: **X Y Z W V** Master Axis name.

axes: **X Y Z W V** Slave axes names.

After this instruction execution, all the selected **Slave axes** will follow exactly the **Master axis** movements, either in point to point or in interpolation mode, counterbalancing any eventual position differences with the master axis itself.

The master-slave chaining is actived only when all the axes are in quiescent status (Q status). If some axis is moving, the instruction waits the end of the movement. In any case, this instruction requires that alls the axes have the same encoder resolution and are positionned at the same absolute quotes when the chain is recalled.

To disable the Chain mode and restore the original operativity, the NORMAL instruction, on the master axis, must be recalled.

See also the instruction: NORMAL.

For instance:

In the following exemple, the X axis (as Master) and the W axis (as Slave) are chained. As showed, all the movement instructions are then referred only to the master axis.

```
PROG

SETQ      X,0
SETQ      W,0
CHAIN     X,W          ;X and W chain enable
VEL       X,10
X         1000,Q
CIRCLE    XY,A,6,10
WEND      XY,Q
X         0,Q
NORMAL    X            ;chain disable
END
```

CIRCLE

Two axes circular interpolation

CIRCLE axes,sense,speed,radius[,revol.]

CIRCLE axes,sense,speed,radius,qf1,qf2

axes: **X Y Z W V**	Selected axes names.
sense: **O A**	Defines: O clockwise sense A counterclockwise.
speed: **x.x**	Working speed value, in mt/1'.
radius: **±x.x**	Radius value.
revol.: **0.25÷15**	Revolutions number.
qf1: **±x.x**	Axis 1 final quote (relative to the starting point)
qf2: **±x.x**	Axis 2 final quote (relative to the starting point)

This instruction performs a two axes circular interpolation: the selected axes must have the *same encoder resolution.*

The **speed** value must be programmed with reference to the vectorial trajectory.

See also the instructions: LINEAR, HELIC, ICNT and ECNT.

Two different programming modes are available:

1) The first mode provides a complete circle, eventualy repeated many times, as defined by the **revol** parameter.

As showed in the following figure, changing the axes sequence and by the sign of the **radius**, it’s possible to program four different dispositions of the circle.

It’s also possible to program, in simplified form, the 1/4, 2/4 or 3/4 fractions of the circumference, inserting the 0.25, 0.5 and 0.75 values in the “revol” parameter.

2) The second mode allows to program any type of circular arc.

In this case only the incremental quotes of the arc final point, with reference to the initial point, and the radius must be programmed.

The radius sign allows to select the =< 180° arc, if positive, and > 180° arc, if negative..

COMP

Two ports contents Compare.

COMP port1,operator,port2[,mask,label]

port1: **000÷255** Address number of the first port to be compared.

operator: **< = >** Compare Operators, may be used also in combination, as, for instance: >= or =<.

port2: **000÷255** Address number of the second port to be compared.

mask: **000÷255** Binary Mask of the bits to be compared

label: Label to branch if the compare result is positive.

This instruction allows to compare the **port1** byte with the **port2,** by considering only the masked bits (if the **mask** parameter is programmed) or the complete byte, if mask is missing. If the result accords to the programmed comparing operator, the program skips the next instruction or jumps to the programmed **label,** if present.

DCOUNT

Counter Decrement

DCOUNT cnt

cnt: **0÷31**	Counter Number (address).

The programmed counter (**cnt**) value is decremented by one.

See also the instruction: SCOUNT.

DELAY

Delay time.

DELAY time

time: **x.x**	Delay Time (value x.x in seconds).

This instruction performs the programmed delay time before to start next instruction execution.

DPTAB

Table Pointer Decrement.

DPTAB axis,table[,label]

axis: **X Y Z W V**	Axis name.
table: **0÷3**	Selected Table number.
label:	Label to branch

This instruction allows to decrement the Pointer of the selected **table** and **axis**: if, after the decrement, the pointer value is zero, the program skips next instruction or branchs to the programmed **label,** if present.

See also the instruction: IPTAB.

ECNT

Contourning path End.

ECNT

Programs the end of a contouring path.

This instruction must be inserted before the last LINEAR, CIRCLE or HELIC instruction of the sequence.

See also the instruction: ICNT.

For instance:

```
PROG

ICNT                                    ;begin contourning path
LINEAR      XYZ,10,100,0,10     ;linear interp. 10 mt/1'
HELICXYZ,A,5,10,-50,3     ;helycoidal interp. 5 mt/1'
ECNT
LINEAR      XYZ,10,100,0,10     ;linear interp. 10 mt/1' (end of path)
WEND        XYZ,Q
END
```

As shown in the exemple, if a 3-axes helycoidal interpolation, in contouring mode, is required, also all the other linear instructions must be programmed as 3D interpolations.

At the end of a contouring path, the WEND axes,Q must be programmed, to wait the axes in quiescent status, because the interpolation instructions release them in F status.

END

End of program.

END

This instruction must be used only at the end of the Main program.

ENDREP

End Repeat.

ENDREP

Closes an instructions block, to be repeated, headed by the REPEAT instruction.

When this instruction is scheduled, the Repeat Counter is decremented and, if not zero, the program returns to the first statement of the block (the first after the Repeat instruction): on the contrary, when the counter is decremented to zero, the repeat block ends and the program follows from the next instruction (immediately after the ENDREP statement).

See also the instruction: REPEAT.

EPROG

End of parallel programs or functions.

EPROG

EPROG num

EPROG program names

EPROG !num

EPROG function names

num: **1÷8**	Parallel programs numbers.
program names:	Parallel programs names.
!num: **!0÷255**	Parallel functions numbers.
functions names:	Paralle functions names.

Used in multitasking mode, ends the selected parallel program or function execution.

If called without parameters, ends the parallel program where it is included.

See also the instructions: APROG, FPROG, RPROG, SPROG, TPROG, WPROG.

For instance:

The program 2 is actived and ended when a Limit switch is detected.

```
PROG                    ; main program
APROG       LSWAIT      ; start the parallel program
X           100,Q
WPROG       LSWAIT      ; wait end of the parallel program
END

PROG        LSWAIT      ; parallel program
WNZ         LIMSW
EPROG                   ; end of parallel  program execution
```

EQTAB

Movement execution to the Table quotes

EQTAB axes,table[,mode]

axes: **X Y Z W V** Selected axes names.

table: **0÷3** Selected Table number

mode: **0 1 2 3** Defines: 0 point to point moving
1 synchronous moving.
2 point to point moving with output driving
3 synchronous moving with output driving

This instruction starts the movements of the selected **axes** assuming the selected **table** values as target positions, with the following operating modes:

mode 0

Point to point movements: for every axis the target position may be considered in absolute or incremental mode.
If in **absolute mode**, the table listed quotes are added to the initial Offset parameter, stored in the head of the table, to allow defining of a local reference point, different to Set Point, if required.
In **incremental mode**, the table listed axes displacement is assumed as incremental, then referred to the actual axes positions.
If more axes are selected, a contemporary starting is performed by the system.

For the other general information about the moving modalities, make reference to the MOV instruction.

mode 1

Synchronous Move: in this mode, every quote value, stored in the table location, is assumed as an incremental displacement must be executed in a fixed time interval, defined by the **rate** parameter as:

interval time = rate * 1 / real time frequency

The axes movements are then synchronized performing any type of path in the space: the effective trajectory then results as a continous sequence of incrementals linear segments, gone along with continous speed.

Table is executed starting from the pointed location until the end.

It's also possible to perform a synchronous movement linking any axis, also if drived by different electronic cards.

As in the point to point movement, execution starts only when all the axes are in F or Q status; on the contrary, the end of the previous movement is waited.

mode 2

Similar to the mode 0, includes also an Output driving, selected by a corresponding location of a parallel output table: this table is pointed by the same index of the quotes tables. Any selected output is actived, if required, *before* the corresponding movements start.

mode 3

Similar to the mode 1, includes the output driving facilities of the mode 2: the selected output are actived at the beginning of the interval time that schedules the corresponding incremental quote location.

For instance:

```
Master                          Slave 1

PROG                            PROG

SPTAB     XY,0,1                SPTAB     XYZ,0,1
SYNC      1      ------->       WSYNC
EQTAB     XY,0,1                EQTAB     XYZ,0,1
WSYNC     1      <-------       SYNC
END                             END
```

In the previous exemple, two tabled synchronous movements, involving two different cards axes, are showed: the master card move the X and Y axes, the slave card move the X, Y and Z axes.

ERROR

Cycle Error

ERROR num[,C]

ERROR num[,W]

num: **1÷255** Cycle error identifying number.

C: **C** If present, the program execution continues also after the error occurs.

W: **W** Similar to the previous, used in particular statistical environ.

Displays the (**num**) stored message related to the Cycle error caused by the program.

The ERROR instruction is used in connection with the Cycle Error file: ERRCYC.

The **num** parametr of the ERROR instruction is, practicaly, the index of the Messages table included in the ERRCYC file.

If the options (**C** or **W**) are missing, the error generating task stops: the execution may be restarted typing the START key.

See also the Section: Cycle Error file.

FCALL

Function execution Call

FCALL num

num: **0÷255** Identifier number of the Function

The program execution follows from the first instruction of the called (**nnn**) Function.

The Function may be called also writing the mnemonic name directly, in the opcode field. In this case neither the FCALL item nor the identifier number must be indicated.

Any Function, called in practice as a subroutine, returns to the calling program by the FRET (Function Return) instruction.

A maximum of 4 nesting levels are admitted.

See also the instruction: FRET and the Section relative to the **Functions File**.

For instance:

```
        PROG

        SETPOINT                ;Set point function execution
LOOP:X  10,Q
        FCALL       30          ;execution of the function N.30
        Y           2,Q
        FCALL       31          ;execution of the function N.31
        BRA         LOOP
```

FPROG

Actives the one or more parallel functions execution, using a free tasks.

FPROG !num

FPROG num,!num

FPROG num,functions names

FPROG functions names

num: **1÷8**	Number of the parallel task.
!num: **!0÷255**	Number of the parallel functions.
functions names:	Parallel functions names.

This instruction, as the APROG, EPROG, SPROG, RPROG, TPROG and WPROG, may be used in multiprograms or in multitasking mode.

Allows to run in execution more non parametric Function in contemporary.

If, in the instruction, the (**num**) number of the tasks are programmed expressly, the corresponding numbers or names of the Functions are strictly associated in the programmed sequence; on the contrary, the parallel functions execution will be started on the first available free tasks. Obviously, if no sufficient tasks are available for functions execution, the system waits until enough tasks are freed, as to allows a simultaneous starting of the all required process.

If a recalled function is already in execution, the system waits its end before to start again.

See also the instructions: APROG, EPROG, RPROG, SPROG, TPROG, WPROG.

FREE

Disables the axes position control loop.

axes: **X Y Z W V**	Selected axes names.

Disables the position control loop for the selected **axes**, leaving them in the open loop (free) status.

This instruction mays be usefully recalled in different cases as, for instance, to drive measure axes during quotes autolearning or to allow an adjust movement when the axis is forced by external mechanical devices. In any case, during the free mode, the axis actual position is periodically monitored to allow, when required, to restore correctly the control loop functionality (by the NORMAL instruction).

It's also possible, in the free status, drive an open loop movement using the VOUT instruction (in this case the movement will be controlled only in velocity), as, for instance, to move the axis towards a mechanical limit switch.

See also the instruction: NORMAL.

For instance:

```
PROG

ABS       X
VEL       X,10
X         1000,Q
FREE      X
VOUT      1,0.1       ;set a 0.1V voltage on the DAC
WIZ       LIMSWT      ;waits the limit swich signal
NORMAL    X
END
```

FRET

Return from Function.

FRET

Allows to exit from the Functon, returning to the calling program. The execution restarts from the instruction following the FCALL statement.

See also the instruction: FCALL.

GAIN

Defines the axis control loop Gain.

GAIN axis,value1[,value2]

axis: **X Y Z W V**	Axis name.
value1: **0.25÷15**	Axis main Gain.
value2: **0.25÷15**	Optional chaining Gain (for Slave axis).

This instruction allows to set the Gain (**value1**) of the position control loop for the selected **axis,** either for point to point or interpolated movements.

The second optional value (**value2**), if programmed, defines a special gain for a secondary control loop, related to the chain feature, allowing the slave axis to move stricly synchronized with the master, avoiding undesired displacement shiftings during the movements.

If this parameter is missing, the defaul value, defined in the Station Setup mode, will be used by the system.

See also the instructions: CHAIN and SETFF.

HELIC

Helycoidal interpolation

HELIC axes,sense,speed,pitch,radius[,revol.]

HELIC axes,sense,speed,pitch,radius,qf1,qf2

axes: **X Y Z W V**	Selected axes name.
sense: **O A**	Defines: O = clockwise A = counterclockwise.
speed: **x.x**	Interpolation velocity [in mt/1'].
radius: **±x.x**	Radius value.
pitch: **±x.x**	Linear displacement of the 3th axis.
revol.: **0.25÷15 I F**	Revolutions number
qf1: **±x.x**	Final target quote for the axis 1 (relative to the starting point)
qf2: **±x.x**	Final target quote for the axis 2 (relative to the starting point)

This instruction performs an Helycoidal Interpolation on the three selected **axes**, must have the same encoder resolution.

The **speed** value must be programmed as vectorial velocity along the tridimensional path.

The first two axes execute a circular interpolation and the third a synchronized linear movement in the direction defined by the pitch parameter sign.

If the optional parameters (letter **I** and/or **F**) are programmed beside the **revol. number** field. the first (I) and/or the final (F) revolutions will be performed only in circular interpolation, *without* the linear movement of the third axis.

For the other geometrical considerations, see the circular interpolation different programming modes.

See also the instructions: LINEAR, CIRCLE, ICNT and ECNT.

For instance:

As showed, the programmed **pitch** relates to the total 3rd axis displacement, executed along the complete three revolutions performed by the instruction.

HTIMER

Stop timer

HTIMER cnt

cnt: **0÷31** Number of the timer.

Suspends the (**cnt**) timer counting, if active: the timer counting value is stopped when this instruction is fetched.

See also the instruction: TTIMER.

For instance:

The following exemple shows a typical use of timer instructions to evaluate the execution time of a procedure.

```
        PROG

        RTIMER   1        ; timer reset
CYCLE:  VCOUNT   0        ; display timer
        STIMER   1        ; start timer
        |
        |
        BRA      CYCLE
```

ICNT

Start Contouring path

ICNT [mode]

mode: **0 1** Defines: 0 slow-down after the interconnection point
1 slow-down before the interc. point

This instruction is used to advise the beginning of a path to be executed with continuity of speed (then without a stop enter a block and another), then a path including a sequence of LINEAR, CIRCLE and HELIC with *tangent continuity*.

The **mode** parameter, valid only if the system is equipped by the HSINT card, allows to select, when the following interpolation blocks have been programmed with decreasing velocity, if the slow-down must be performed before to reach the interconnection point (mode = 1), as to begin the new block already with the programmed speed, or to begin slow-down after the interc. point, making them in the new block (mode = 0).

Independently of the mode parameter, in the case of following blocks with growing-up speed, the acceleration ramp is alwais started in corresponce of the interconnection point.

If the HSINT card is not supplied, the change of speed are executed alwais in the interconnection point, without ramps.

See also the instruction: ECNT.

ICOUNT

Counter Increment

ICOUNT cnt

cnt: **0÷31** Counter identifying number.

The (**cnt**) counter value is incremented by one.

See also the instruction: SCOUNT.

INC

Defines, for the selected axes, the Incremental programming mode.

INC axes

INC ACaxe

axes: **X Y Z W V**	Names of the DC axes.
ACaxe: **X1÷U3**	Names of the AC axes.

After this instruction, alls the point to point programmed displacements, for the selected **axes,** are assumed in incremental mode.

See also the instruction: ABS.

INCOUNT

Copy of the counter content in a port/ports

INCOUNT cnt,portl[,porth]

cnt: **0÷31**	Counter number.
portl: **000÷255**	Destination port address (for Least significant Byte)
porth: **000÷255** [optional]	Destination port address (for Most significant Byte)

This instruction copies the **(cnt)** counter contents in the following mode: the LOW byte in the **portl** and the HIGH byte in the **porth**, this last only if programmed.

See also the instruction: OUTCOUNT.

INP

Copies a port contents in another

INP port1,port2[,mask]

port1: **000÷255**	Source port address.
port2: **000÷255**	Destination port address
mask: **000÷255**	Bit Mask to be copied (significant bits = 1).

Copies the **port1** contents in the **port2** , allowing to transfer, eventualy, only a part of the bits according to the programmed **mask** parameter.

If the mask is missing, the 255 value (corresponding to FF Hex.) is assumed by default, enabling all bits copying.

IPTAB

Table Pointer increment

IPTAB axis,table[,label]

axis: **X Y Z W V**	Axis names.
table: **0÷3**	Table number
label:	Label to Jump.

This instruction increments the **table pointer** relative to the selected **axis:** when the maximum number of table locations is overcame, program skips next instruction or jumps to the selected **label,** if programmed**.**

See also the instruction: DPTAB.

LIMITOFF

Axes SW limit switch control Disable

LIMITOFF axes

axes: **X Y Z W V**	Selected axes names

Disables the positive and negative limits control, for the selected **axes**.

The limit-quotes must be defined in the **Staton Setup** mode.

See also the instruction: LIMITON.

LIMITON

Axes SW limits switches control Enable

LIMITON axes

axes: **X Y Z W V**	Selected axes names

Enables the positive and negative limits control, for the selected **axes**.

The limit-quotes must be defined in the **Staton Setup** mode.

See also the instruction: LIMITOFF.

LINEAR

Linear Interpolation

LINEAR axes,speed,qf1,qf2[,qf3]

axes: **X Y Z W V**	Selected axes names
speed: **x.x**	Interpolation velocity [in mt/1'].
qf1: **±x.x**	Incremental displacement for the axis 1.
qf2: **±x.x**	Incremental displacement for the axis 2.
qf3: **±x.x**	Incremental displacement for the axis 3.

This instruction executes a linear interpolation on the two or three selected **axes**, must have the same encoder resolution.

The **speed** value must be programmed according to the vectorial moving direction.

The (**qf1, qf2, qf3**) quotes must be programmed in incremental format, refered to the actual position of the axes.

See also the instructions**:** CIRCLE, HELIC, ICNT and ECNT

MESSAGE

Indexed message display.

MESSAGE num[,P]

MESSAGE num[,row]

num: **1÷255**	Number of the message to be displayed.
P: **P**	Parameter for message Disk storing option
row: **1÷8**	Row number where the message must be displayed.

Allows to display on the screen the **num** message, and, to store them (option **P**) on the disk.

If the **P** option is selected, and in the Station Setup the **VIDEO message Saving** option enabled, when the PC receives the message, stores them on the disk in the REPORT file.

By the **row** parameter, user can select the displaying field on the screen: in the case of incorrect value (not included enter 1 and 8) the message will be displayed on the row n. 1.

See also the instruction: VIDEO.

MOV

Axes point to point movement start.

MOV axis,quote[,status]

axis quote[,status]

axis1 quote,axis2 quote,...[,status]

MOV ACaxis,quote[,status]

ACaxis quote[,status]

ACaxis1 quote,ACaxis2 quote,...[,status]

axis: **X Y Z W V**	Selected axes names.
quote: **±x.x**	Final target Quote.
status: **A R D F Q**	Status to be waited.
ACaxis: **X1÷U3**	Selected AC axes names.

Starts, in contemporary, the selected axes.

The **quote** parameter may assume a different meaning according to the selected movement typology:

incremental defines the signed displacement from the axis actual position.

absolute defines the absolute (with reference to the SetPoint or Working Zero position) target quote (see also the quotes correction instruction as : SETOFS, USEOFS).

If the **status** parameter is programmed, defines the axis status to be waited before the next instruction fetching (see also the WEND instruction).

If an axis is already moving, the instruction waits the F status before to start them again.

For the AC axes moving, the X1 ta U1 axes may be selected directly omitting the 1 number, then writing simply X, Y,..and so on.

Exemples:

1) axes movements:

```
PROG

ABS         XY          ; only if DC motor
VEL         X,10        ; only if DC motor
VEL         Y,15        ; only if DC motor
MOV         X,20
MOV         Y,20
WEND        XY,Q
END
```

2) contemporary axes movement

```
PROG

VEL         X,10        ; only if DC motor
VEL         Y,10        ; only if DC motor
VEL         Z,10        ; only if DC motor
X 100,Y 200,Z 10,Q
END
```

3) Axis status synchronisation

```
PROG

VEL         X,10        ; only if DC motor
X           100,D       ; wait slow down status
SET         HYDRAM      ; drive hydraulic ram
END
```

MOVOFS

Point to point movement Start (target position defined by an OFFSET register).

MOVOFS axes,offset

axes: **X Y Z W V**	Selected axes names.
offset: **0÷31**	Offset Register number

This instruction starts, in contemporary, alls the selected axes, towards the target positions defined by the selected Offset Register.

For the other consideration, make reference to the MOV instruction.

See also the instructions: SETOFS, USEOFS, CALOFS, ADDOFS, SUBOFS, TOFS.

For instance:

```
PROG

ABS       X
VEL       X,10
SETOFS    X,0,1000
SETOFS    X,1,100
ADDOFS    X,0,X,1,X2
MOVOFS    X,2          ;moves X axis to the Offset Reg. n. 2 quote
WEND      X,Q
END
```

MQOFS

Copy a quote into an Offset Register

MQOFS axis1,table,axis2,offset[,num]

MQOFS axis1,type,axis2,offset[,num]

MQOFS axis1,@offset,axis2,offset[,num]

axis1: **X Y Z W V**	Source axis name.
table: **0÷3**	Source Table number.
type: **R T**	Defines: R real quote T theorical quote.
@offset: **@0÷@31**	Source Offset Reg. Number
axis2: **X Y Z W V**	Destination axis name
offset: **0÷31**	Destination Offset Register number
num: **0÷15**	Number of the station where the quotes to be copied

This instruction may be used to save, into an Offset Register, an autolearned quote, during an automatic working cycle.

It’s possible to copy a **Real (R)** or a **Theorical (T)** quote of an axis, or a **Table location value** or, eventualy, another **Offset Register**, also relative to a different Axes or **Station**. In the case of different axes, is compulsory they have the same encoder resolution.

The source quote to be copied is calculated as:

type = R or T: to the real or theorical quotes are subtracted the eventual selected offset.

Source Table: to the pointed quote is added the main Table Offset.

See also the instructions: MQTAB, SETOFS, USEOFS, CALOFS, ADDOFS, SUBOFS, TOFS, MOVOFS.

MQTAB

Copy a quote in a Table.

MQTAB axis1,table1,axis2,table2[,num]

MQTAB axis1,type,axis2,table2[,num]

MQTAB axis1,@offset,axis2,table2[,num]

axis1: **X Y Z W V**	Source axis name.
table1: **0÷3**	Source Table number.
type: **R T**	Defines: R real quote T theorical quote.
@offset:**@0÷@31**	Source Offset Register number
axis2: **X Y Z W V**	Destination axis name.
table2: **0÷3**	Destination Table number.
num: **0÷15**	Destination Station number.

Normaly this instruction may be used to store, into a Table, a group of quotes detected during an automatic autolearning cycle.

It’s possible to copy an actual axis quote, **Real (R)** or **Theorical (T),** or a complete quotes Table, also relatives to different axes or stations, in this case, compulsory, with the same encoder resolution.

The source quote is calculated as:

type = R or T: the eventual selected offset is subtracted to the actual (R or T) quote before storing.

source table : to the pointed quotes are added the main table offset.

The destination quote is saved with the following operation: the main offset of the destination table is subtracted from the calculated quote to be stored.

See also the instruction: MQOFS.

NOP

No operation

NOP

No operation is made by this instruction.

NORMAL

Axes position control loop Enable.

NORMAL axes

axes: **X Y Z W V** Selected axes names.

This instruction restores the position control loop for the selected axes, disabling the Free or the Chain command.

This is a normal condition setted, by default, at the power up of the system. In any case, user is suggested to insert this instruction, at the beginning of the programs or the Set Point procedure, to recover any possible emergency situations have been disabled the axes control loop.

See also the instructions: FREE and CHAIN.

For instance:

```
NORMAL     XZ
```

NOT

Binary logical NOT

NOT op[,result]

op: **bitport**	Operand.
result: **bitport**	Result.

Executes the logical NOT operation on the **op** port, storing the result in the same port or in the **result** port, if programmed.

OR

Binary logical OR on two bytes (bit per bit OR).

OR op1,op2[,result]

op1: **bitport**	Operand 1.
op2: **bitport**	Operand 2.
result: **bitport**	Result.

This instruction executes the logical bit per bit OR enter the **op1** and **op2** operanding, storing the result in op2 or in the **result** port, if programmed.

OUT

Writes a data into a port

OUT dataport[,mask]

data: **000÷255** Binary Data to be writed.

port: **000÷255** Destination Port.

mask: **000÷255** Decimal value of the Binary Mask to select the bits of the
port to be writed.

This instruction executes a **data** writing on a output **port**, allowing to select only a group of bits to be involved in the operation. In this mode, it's possible to set or reset many bits in contemporary, allowing a more speedy execution.

For instance:

* File of I/O Definitions

```
EV01        0008        ; bit 0 (mask = 1)
EV02        1008        ; bit 1 (mask = 2)
EV10        4008        ; bit 4 (mask = 16)
EV12        7008        ; bit 7 (mask = 128)
```

* Program

```
    PROG
    OUT         001008,147    ; set EV01, reset EV02,EV10,EV12
    END
```

The mask = 147 is the decimal value corresponding to the binary mask bit.

OUTCOUNT

Copy the input port/s contents in a Counter.

OUTCOUNT cnt,portl[,porth]

cnt: **0÷31**	Destination Counter Number.
portl: **000÷255**	Low byte source port address
porth: **000÷255**	High byte source port address.

Copies the **portl** and **porth** contents in the **(cnt)** 16-bit counter and, precisely: the **portl** contents in the low byte and, if programmed, the **porth** contents in the high byte.

If **porth** is missing, in the counter high byte all zeros are entered.

See also the instruction: INCOUNT.

PAUSE

Suspend the program or task execution.

PAUSE

The suspended task or program execution may be restarted typing the START key.

If used in a PLC Function, this instruction will be ignored.

REPEAT

Instructions block Repeat

REPEAT num

REPEAT #cnt

num: **1÷65535** Block Repeat direct Number.

#cnt: **#0÷#31** Counter Address which contents is the Repeat number.

Repeats the block of instructions included enter the REPEAT and the ENDREP statement: the Repeat number may be programmed directly, by the (**num**) parameter, or in indexed mode, by the contents of the selected (**#cnt**) counter.

The maximum number of REPEAT/ENDREP loops nesting is 4.

See also the instruction: ENDREP.

For instance:

Two nested repetition blocks:

```
PROG

REPEAT    5     ;1st repeat.                              |
SET       PST   ;set out                                  |
REPEAT    10    ;2nd repeat.          |                   |
RES       PST   ;reset out            | BLOCK 2           | BLOCK 1
ENDREP          ;end of the 2nd loop. |                   |
ENDREP          ;end of the 1st loop.                     |
END
```

As shown, the 2nd block is executed 10 times (REPEAT 10), whereas the first block 5 times (REPEAT 5); then, since the second block is nested in the first, will be repeated 50 times (10*5).

RES

Reset of a bit of an output port

RES bitport

bit: **0÷7**	Bit to be resetted.
port: **000÷255**	Output Port address.

The **bit** of the output **port** is cleared (status = 0).

The bitport argument may be also defined in mnemonic format, according to the symbolic name setted in the I/OR Definition file.

See also the instruction: SET.

RESTART

Firmware subroutine Call

RESTART num

num: **0÷255** Firmware subroutine number.

A Firmware procedure, identified by the (**num**) parameter, is called in execution.

This instruction typology may be executed only if the corresponding firmware procedure, normaly customized, is supplied by the system. These procedures are installed, if required, for special purpose procedures, according to the needs of the machine.

RET

Return from subroutine

RET

Allows to exit from the subroutine, returning to the calling program; the execution restarts from the instruction following the CALL statement.

See also the instruction: CALL.

RPROG

Restarts execution of one or more parallel programs or functions.

RPROG num

RPROG program names

RPROG !num

RPROG functions names

num: **1÷8**	Number of the parallel tasks.
program names:	Parallel programs names.
!num: **!0÷255**	Parallel Functions number.
functions names:	Parallel Functions names.

Used in multitasking mode, allows to restart the selected rograms or functions execution.

The execution restarts from the instruction following the has been suspended by the SPROG instruction.

See also the instructions: APROG, EPROG, FPROG, SPROG, TPROG, WPROG.

RTIMER

Timer Reset

RTIMER cnt

cnt: **0÷31** Timer number.

Clears the selected (**cnt**) timer contents.

See also the instruction: TTIMER, HTIMER, STIMER.

For instance:

This procedure shows as a timer may be used to evaluate a Cycle time.

```
            PROG

            RTIMER    1           ;reset timer
CICLO:      VCOUNT    0           ;display timer
            STIMER    1           ;start timer
            |
            |
            BRA       CICLO
```

SCOUNT

Loads a value in a Counter

SCOUNT cnt1,value

SCOUNT cnt1,#cnt2

cnt1: **0÷31**	Counter number.
value: **0÷65535**	Value to be loaded into a counter.
#cnt2: **#0÷#31**	Number of the source counter (where the value may be unloaded)

Loads the (**cnt1**) counter with a **value** defined or directly by the instruction argument or by the selected (**#cnt2**) counter contents.

All the 32 counters (0÷31) may be used for this operation.

This instruction is normaly used in connection with the:

ICOUNT, DCOUNT, TCOUNT, VCOUNT.

Typical uses of the counters registers is the working pieces or the error conditions counting, during the operative cycles.

See also the Section relative to the **Counters Description,** in the **General Module Setup** option.

SERVO

Set a limit value of the position error, to detect an axis servoerror.

SERVO axis[,quote]

axis: **X Y Z W V**	Axis Name.
quote: **x.x**	Servoerror limit.

This instruction allows to modify the axis servoerror limit, setted, by default, equal ±2047 encoder pulses (±20.47 mm for an axis with 0.01 mm resolution).

This limit defines the maximum position error, as difference enter the theorical and the actual quote of the moving axis, may be accepted by the system. Since this value normaly may change in function of the speed, resolution and gain regulation of the servo system loop, the default value may be altered by the present instruction.

In any case, the servoerror limit cannot overcome the **32768/(Gain*4)** pulses and any other greater values are automaticaly limited by the system.

Il the value parameter is missing, automaticaly the default value is entered.

SET

Set of a bit of an output port

SET bitport

bit: **0÷7**	Number of the bit to be setted.
port: **000÷255**	Output port address.

The selected **bit** of the output **port** is setted in the logical 1 status.

The bitport argument may be defined in mnemonic format, according to the I/OR Definition file.

See also the instruction: RES.

Exemples:

1) Exemple of symbolic definition procedure:

```
* I/OR Definition File
;Inputs
   FCPIST       0000         ;hydr. ram limit switch
;Outputs
   HYDRRAM      0008         ;hidr. ram actuator

* Program
     PROG
     SET        HYDRRAM      ;hydr ram UP
     WIZ        FCPIST       ;waiting for UP limit switch ON
     END
```

2) the same procedure could be writed as:

```
     PROG
     SET        0008         ;hydr ram UP
     WIZ        0000         ;waiting for UP limit switch ON
     END
```

SETFF

Axis feed forward Enable

SETFF axis,0

SETFF axis[,num1/num2]

axis: **X Y Z W V**	Axis name.
num1: **0.25÷15**	Speed multiplier for the feed forward computing
num2: **1 2 4 8**	Speed divisor for the feed forward computing.

This instruction allows to active the axis **feed forward** value for a point to point movements.

If a **zero (0)** value is entered, the feed forward contribution to the position control loop is cleared.

If no value is programmed, the default value, defined in the Station Setup file, will be automaticaly assumed.

See also the instruction: GAIN.

SETOFS

Axis Offset Register value set

SETOFS axis,offset,quote

axis: **X Y Z W V**	Axis name.
offset: **0÷31**	Offset number.
quote: **±x.x**	Offset Quote.

Loads, in the selected **offset** register of the **axis,** the programmed **quote:** this instruction may be used to move the machine Zero point, defined by the Set point procedure. As a matter of fact, once this quote is recalled for correction by the USEOFS instruction, this value will be added to the programmed target quotes in the absolute movements.

The offset registers may also be used, in general, to store autolearned quotes, to make calculations, to move axes, and so on..

See also the instruction: USEOFS, CALOFS, ADDOFS, SUBOFS, TOFS, MOVOFS, MQOFS.

For instance:

In the following exemple, a correction quote is loaded and recalled by the USEOFS instruction. In the following programmed displacement at the target quote (100), the axis will be positionned to the required position added to the offset of correction (value 1000), the to the effective quote 1100.

```
PROG

SETOFS      X,0,1000
USEOFS      X,0
VEL         X,10
X           100,Q
END
```

SETPF

Axis actual position reset from external signal.

SETPF axis,bitport,status

axis: **X Y Z W V**	Axis name.
bit: **0÷7**	Bit number.
port: **000÷255**	Address port.
status: **0 1**	Defines: 0 bitport active if closed 1 bitport active if open

This special instruction allows to clear the **axis** real quote when the selected signal, defined by the **bitport** address, switches in the active **status.**

It may be used to execute a zero-piece or a normal Set point procedure.

During this instruction execution, the Input lines sample cycle is reduced to 1 millisecond of period, rather than the usual 5 milliseconds, to allow a highter speed of setpoint search according to the precision of detection required (i.e: 6 mt/sec with 0.1 mm of resolution).

It’s suggested, in any case, to start this instruction one axis at the times.

SETQ

Set of the axis real quote.

SETQ axis,quote

SETQ ACaxe,quote

axis: **X Y Z W V**	Axis name.
quote: **±x.x**	Quote value to be entered.
ACaxe: **X1÷U3**	Axis name.

This instruction is usually recalled to clear the axes **quotes** during the Set point or Zero piece procedure. If the axis is moving, it is stopped and placed in the "Q" status.

In any case, the new quote to be entered may be anyone, not compulsory the zero value.

For instance:

```
PROG

VEL        X,0.1
X          1000
WNZ        TC0X        ;wait encoder zero mark
SETQ       X,0
END
```

SHIFTL

Rotate left of a port

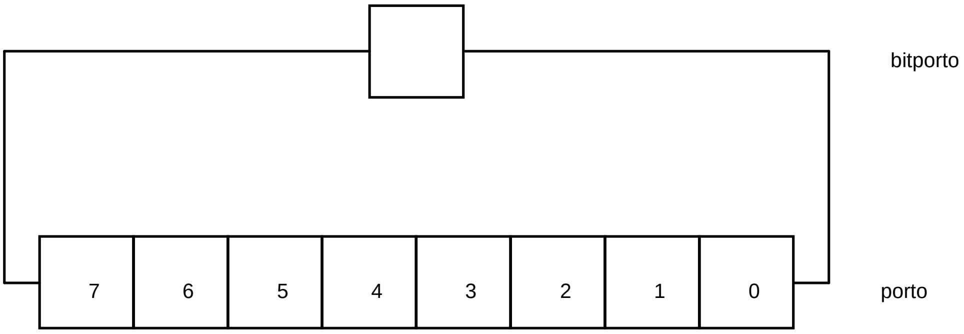
SHIFTL port,bitport

port: **000÷255** Port address.

bit: **0÷7** Bit number.

This instruction executes a left rotate of the programmed **port** contents. The specified **bitport** shifts in the bit 0 of the port and the bit 7 of the port enters in the specified bitport.

The rule of the port rotate process is shown in the following drawing:



This rule allows to chain multiple shifts over more ports.

See also the instruction: SHIFTR.

SHIFTR

Rotate right of a port

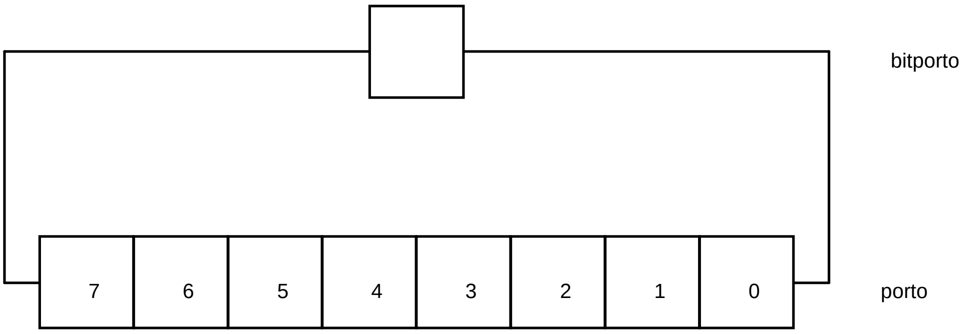
SHIFTR port,bitport

port: **000÷255** Port address.

bit: **0÷7** Number of a bit.

Execute a right rotate of the selected **port**. The programmed **bitport** (second argument) shifts into the bit 7 of the port, the bit 0 of the port enters in the specified bitport.

The rotate process is shown in the following drawing:



See also the instruction: SHIFTL.

SKE

Skip on axis status/direction or interpolation status.

SKE axes,status[,label]

SKE axes,direction[,label]

SKE ACaxe,status[,label]

SKE ACaxe,direction[,label]

SKE interpolation,status[,label]

axes: **X Y Z W V**	Axes name
status: **A R D F Q**	Axes status.
direction: **+ -**	Direction.
label:	Label to branch
ACaxe: **X1÷U3**	Axes name.
interpolation: **I**	selects test on interpolation.

This instruction compares the selected **axes status** or **direction** with his argument and, if equals, skips next instruction or jumps to the selected **label,** if present.

In the case of **interpolation status** test, the possible status to be compared are: **R**, **D**, **F**.

For instance:

```
            PROG
            VEL         X,10
            X           1000
LOOP:       SNZ         FC,FCON
            SKE         X,Q,FINE
            BRA         LOOP

FCON:       STOP        X
            ERROR       1
            END

FINE:       VIDEO       END OF MOVEMENT
            END
```

SKZ

Skip if (bitport) is zero

SKZ bitport[,label]

bit: **0÷7**	Bit Number.
port: **000÷255**	Address of the port
label:	Label to jump.

This instruction tests the (**bitport**) line and skips next instruction or branches to the **label** (if present) if zero.

The bitport argument may be defined in symbolic format, according to the I/OR Definitions file.

See also the instruction: SNZ.

Exemples:

1) test with next intruction skip

```
            PROG
            SET       HYDRRAM
            SKZ       FCPIST
            BRA       ACTIVE
            VIDEO     LIMIT SWITCH DISACTIVE
            END

ACTIVE:     VIDEO     LIMIT SWITCH ACTIVE
            END
```

2) test with branch to a label

```
            PROG
            SET         HYDRRAM
            SKZ         FCPIST,DISATT
            VIDEO       LIMIT SWITCH ACTIVE
            END

DISATT:     VIDEO       LIMIT SWITCH DISACTIVE
            END
```

SNZ

Skip if (bitport) is not zero.

SNZ bitport[,label]

bit: **0÷7**	Bit number
port: **000÷255**	Port address.
label:	Label to jump.

As to the previous instruction, but the skip/branch condition is the (**bitport**) equal to “1”.

See also the instruction: SKZ.

SPECIAL

Special firmware program execution.

SPECIAL num

SPECIAL #cnt

num: **1÷255**	Number of the special program to be started.
#cnt: **#0÷#31**	Number of the counter for indexed selection of the special program.

This instruction actives a special program identified by the direct (**num**) argument or by the (**#cnt**) counter contents.

This instruction may be used only if the corresponding firmware procedures, normaly customized according to the machine requirements, are loaded into the cards memory.

SPROG

Suspends the parallel program or function execution.

SPROG

SPROG num

SPROG program name

SPROG !num

SPROG functions name

num: **1÷8**	Number of the parallel program
program name:	Name of the parallel program
!num: **!0÷255**	Number of the parallel function
functions name:	Name of the parallel function

Used in multiprogramming mode, suspends the parallel program or function execution.

In this case, the selected program or function, ends the current instruction and stops. The execution may be restarted by the RPROG instruction.

If this instruction is recalled without argument, suspends its own program.

See also the instructions: APROG, EPROG, FPROG, RPROG, TPROG, WPROG.

SPTAB

Table Pointer set.

SPTAB axes,table,num

SPTAB axes,table,#cnt

axes: **X Y Z W V**	Axes names.
table: **0÷3 A**	Number of the table: if 0÷3 is the selected table if A (All) alls the tables.
num: **1÷8000**	Number of quotes location.
#cnt: **#0÷#31**	Number of the counter containing the number of the quotes location

This instruction sets the **axes** pointer of the selected **table** (if the (0÷3) number is programmed) or of the all (if **(A)**), to the location defined or by the direct **num** argument or, in indexed mode, by the programmed (**#cnt**) counter contents.

If programmed directly, the **num** value must be included enter 1 and the maximum length defined for the table. This limit must be taken in account also if a (#0÷#31) counter is used to load the table pointer.

For instance:

This exemple shows two tables (0 and 1) using to define a pallet charge and discharge areas, with two different numbers of quotes, defined in the Table dimensioning.

```
            PROG

            SCOUNT      0,0         ;reset counter of loading pallet
            SCOUNT      1,0         ;reset counter of unloading pallet
            VCOUNT
            SPTAB       XY,A,1      ;reset tables pointers.
CICLOP:     VIDEO       ..LOADING..
            EQTAB       XY,0        ;load starting position
            Loading                 ;Loading function
            IPTAB       X,0         ;Load table pointer increment
            NOP
            IPTAB       Y,0         ;end of table ?
            BRA         CICLOD      ;no
            VIDEO       LOADING PALLET AVOID
            ICOUNT      0           ;loading pallet counter increment
            VCOUNT
            ChangeLoadPall                ;function for loading pallet change
            SPTAB       XY,0,1      ;reset Loading Table pointer
CICLOD:     VIDEO       ..UNLOADING..
            EQTAB       XY,1        ;unload positioning
            Unloading               ;function for unloading
            IPTAB       X,1         ;Unloading table pointer increm.
            NOP
            IPTAB       Y,1         ;end of table ?
            BRA         CICLOP      ;no
            VIDEO       UNLOAD PALLET COMPLETED
            ICOUNT      1           ;unload. pallets counter increm.
            VCOUNT
            ChangeUnldPal           ;Unload pallet change function
            SPTAB       XY,1,1      ;unload. table pointers reset
            BRA         CICLOP
```

STIMER

Start timer

STIMER cnt

cnt: **0÷31** Number of the timer.

Starts the programmed (**cnt**) timer counting, incremented every 0.01 seconds; the maximum counting is then 655.35 seconds.

If the counting has been stopped by the HTIMER instruction, this instruction restarts the timer from its current value.

See also the instruction: TTIMER, HTIMER and RTIMER.

For instance:

This exemple shows as a timer may be used to display the process cycle time.

```
            PROG

CICLO:      VCOUNT      0           ;displying the  timer
            RTIMER      1           ;timer reset
            STIMER      1           ;timer start
            |
            |
            BRA         CICLO
```

STOP

Program or axes stop.

STOP

STOP axes

STOP interpolation

STOP ACaxe

axes: **X Y Z W V**	Axes names.
interpolation: **I**	Selects interpolation.
ACaxe: **X1÷U3**	AC axis names.

When used without arguments, the **STOP** istruction has the same effect as the STOP button in Automatic mode and precisely: all the moving axes slow down and stop and the program execution is suspended. By pushing the START button the execution and the axes movements restart.

If used as **STOP axes**, only the moving axes (then in status (A) or (R)) are stopped, with slowing down.

The **STOP I** instruction suspend the interpolated movements, leaving the axes in point to point manegement.

For instance:

```
PROG

ABS       X
VEL       X,10
X         1000
WNZ       FC         ; waiting the limit switch
STOP      X          ; axis stop
END
```

SUBOFS

Two axis Offset registers Subtract

SUBOFS axis1,offset1,axis2,offset2[,axis3,offset3]

axis1: **X Y Z W V**	Source Axis 1 name
offset1: **0÷31**	Source Offset register 1 number
axis2: **X Y Z W V**	Source Axis 2 name
offset2: **0÷31**	Source Offset register 2 number
axis3: **X Y Z W V**	Destination Axis name.
offset3: **0÷31**	Destination Offset register number

This instruction subtracts from the **offset1** register quote the **offset2** register quote, storing the result in the **offset3** register, if programmed, or in the offset2, on the contrary.

The subtraction may involve two different axes, must have the same encoder resolution.

See also the instructions: SETOFS, USEOFS, CALOFS, SUBOFS, TOFS, MOVOFS.

SYNC

Synchronisme Send

SYNC A

SYNC num

A: **A** Synchronisme sended to the all stations.

num: **0÷15** Number of the station to be synchronized

Allows to synchronize the station 0 (Master) with one (**num**) or all (**A**) Slaves stations or to synchronize a Slave station with the Master.

This instruction assumes a different meaning if used on a Master or on a Slave station:

- **on a Master station**: synchronizes the selected Slave station: on this last must be programmed the relative WSYNC (wait synchronisme) instruction. The functionality is the following:

SYNC (station start)

Master --> Slave

RDY (station ready) end of synchronisme

Slave --> Master

- **on the Slave station**: synchronizes the Master station, where must be present the WSYNC num (wait from the num station), with the following functionality:

RDY (station ready) end of Slave synchronisme

Slave --> Master end of Master synchronisme

The SYNC and RDY signals are defined by the system and do not require hardware phisical Input/Output lines.

If used on the station 0 may be:

```
SYNC   A
SYNC   num
```

If used on the Slave stations may be:

```
SYNC   ( 0 is assumed by default)
SYNC   0
```

See also the instruction: WSYNC.

For instance:

```
Master           Slave 1

PROG             PROG

SYNC 1  ------>  WSYNC                        ; start operation
|                X           100,Q
|                SET         0008
|                X           0,Q
WSYNC 1 <---     SYNC                         ; end operation
END              END
```

TCOUNT

Counter contents Test

TCOUNT cnt1,operator,value[,label]

TCOUNT cnt1,operator,#cnt2[,label]

cnt1: **0÷31**	Number of the counter.
operator: **< = >**	Compare Operator: may be also a combination of operators as, for instance: >= or =<.
value: **0÷65535**	Comparing Value
#cnt2: **#0÷#31**	Number of the counter containing the comparing value.
label:	Label to branch if the test is verified.

Compares the (**cnt1**) counter contents with the direct inserted **value** or with the (**cnt2**) counter contents, if programmed. If the result accords to the programmed comparing **operator**, the program skips next instruction or branches to the **label,** if present**.**

See also the instruction: SCOUNT.

TOFS

Two axis offset register Compare

TOFS axis1,offset1,operator,axis2,offset2[,label]

TOFS axis1,offset1,operator,axis2,type[,label]

axis1: **X Y Z W V**	First axis name.
offset1: **0÷31**	First axis Offset number
operator: **< = >**	Compare Operator: may be also a combination of operators as, for instance: >= or =<.
axis2: **X Y Z W V**	Second axis name
offset2: **0÷31**	Second axis Offset number
type: **R T**	Defines: R real quote T theorical quote.
label:	Label to branch.

Compares the **axis1 offset1** contents with the **axis2 offset2 contents** or with the **axis2 Real (R)** or **Theorical (T)** quote.

If the result accords with the operator, the program skips next instruction or branches to the **label**, if present.

See also the instructions: SETOFS, USEOFS, CALOFS, ADDOFS, SUBOFS, MOVOFS.

TPROG

Test if the selected parallel program or function is in execution.

TPROG num[,label]

TPROG program name[,label]

TPROG !num[,label]

TPROG function name[,label]

num: **1÷8**	Number of parallel programs
label:	Label to branch
program name:	Parallel program name.
!num: **!0÷255**	Number of parallel function.
function name:	Parallel function name.

Used in multiprogramming mode, this instruction allows to test if one or more **parallel programs** or **functions** are in execution or not. In negative case, the program skips next instruction or jumps to the **label**, if programmed.

See also the instructions: APROG, EPROG, FPROG, RPROG, SPROG, WPROG.

For instance:

```
         PROG
         SET       HYDRRAM           ;actives hydrram
         STIMER    1
         APROG     WAITING           ;start parallel program
LOOP:    TTIMER    1,>,10,ERROR
         TPROG     WAITING           ;prog. WAITING in execution ?
         BRA       LOOP              ;yes
         END                         ;no
ERRORE:  VIDEO     LIMIT SWITCH NOT DETECTED
         END
         PROG      WAITING           ;parallel program
         WIZ       LIMSWRAM          ;waits limit switch hydrram
         EPROG
```

TPTAB

Table pointer value Test

TPTAB axis,table,operator,num[,label]

TPTAB axis,table,operator,#cnt[,label]

axis: **X Y Z W V**	Axis name.
table: **0÷3**	Table Number
operator: **< = >**	Compare Operator: may be also a combination of operators as, for instance: >= or =<.
num: **1÷8000**	Number of quotes location.
#cnt: **#0÷#31**	Number of counter storing the number of quotes location
label:	Label to jump

Compares the **table** pointer with the programmed (**num**) value or with the (**#cnt**) counter contents. If the result of the compare accords with the programmed **operator,** the program skips next intruction or jumps to the **label**, if present.

TQ

Axis quote Test

TQ axis,operator,quote,type[,label]

TQ ACaxe,operator,quote,R[,label]

axis: **X Y Z W V** Axis name.

operator: **< = >** Compare Operator: may be also a combination of operators as, for instance: >= or =<.

quote: **±x.x** Quotes to compare.

type: **R** or **T** Type di quote: R selects the axis real quote
T selects the axis theorical quote

label: Label to jump.

ACaxe: **X1÷U3** **AC**Axis name.

Compares the **Real** (R) or **Theorical** (T) **axis quote** with the programmed value. If the compare result accords with the selected **operator,** the program skips next instruction or jumps to the **label**, if present.

The compare operator works with signed numbers, then, for instance, the +100 is considered greater then -10000. Concerning the AC axes, only the real quote, read from the encoder hardware counter is used, then the **R/T** parameter will be, in this case ignored.

For instance:

In the following exemple, the Y axis movement is executed only when the X axis theorical quote is >=50.

```
            PROG
            SETQ        X,0
            VEL         X,10
            X           1000
LOOP:       TQ          X,<,50,T,LOOP       ; wait until the theorical X quote is <
   50
            Y           100,Q
            END
```

TQTAB

Compares an axis quote with a value in the table

TQTAB axis,table,operator,type[,label]

axis: **X Y Z W V** Axis name.

table: **0÷3** Table number.

operator: **< = >** Compare Operator: may be also a combination of operators
as, for instance: >= or =<.

type: **R T** Defines: R Real quote
T Theorical quote.

label: Label to jump.

Compares the **Real** (R) or **Theorical** (T) **axis quote** with the current location of the programmed **Table**. If the compare result accords with the selected **operator,** the program skips next instruction or jumps to the **label**, if present.

The compare operator works with signed numbers, then, for instance, the +100 is considered greater then -10000.

TTIMER

Timer value Test

TTIMER cnt,operator,time[,label]

cnt: **0÷31**	Timer Number.
operator: **< = >** as, for	Compare Operator: may be also a combination of operators instance: >= or =<.
time: **x.x**	Time (in seconds).
label:	Label to jump.

Compares the (**cnt**) timer contents with the programmed **time** value. If the result accords with the selected **operator**, the program skips next instruction or jumps to the **label**, if present.

See also the instructions: STIMER, HTIMER, RTIMER

USEOFS

Axis offset Call

USEOFS axes,offset

axes: **X Y Z W V**	Selected axes names.
offset: **0÷31**	Offset number.

Recalls one of the **Offset** registers previously entered by the SETOFS instruction.

Then, once recalled by the USEOFS instruction, the Offset value will be alwais added to the final target positions, programmed in the absolute deplacements.

See also the instruction: SETOFS, CALOFS, ADDOFS, SUBOFS, TOFS, MOVOFS.

VCOUNT

Counters/Timers block Display

VCOUNT   num

VCOUNT   #num

num: **0÷3**	Block of counters :	0 = counters 0÷7 1 = counters 8÷15 2 = counters 16÷23 3 = counters 24÷31
#num: **0÷31**	Number of the counter	

Sends to PC the programmed (**num**) counters block contents.

If the **#num** argument is used, only the specified counter contents is sended to PC.

The PC displays only the contents of the counter or timer defined in the **Module General Setup** , in the **Counters description** option, if enabled.

See also the instruction: SCOUNT.

VEL

Axis Speed selection

VEL axis[,value]

VEL interpolation[,value]

axis: **X Y Z W V**	Axis name.
value: **x.x**	Speed value (in mt/1').
interpolation: **I**	Selects interpolation.

After this instruction, all the selected axis movements will be executed with the selected new **speed** value, if less or equal to the maximum Setup value; otherwise it will be automaticaly limited to this value..

The programmed value corresponds to the steady speed after the acceleration ramp: this value may be, in any case, modified also during the axis movement.

If the numerical control is equipped by the optional **HSINT** card, this instruction allows to limit also the interpolation vectorial velocity. If missing, also in this case, the Setup value will be assumed by default.

Exemples:

1) point to point movement

programmed speed

0

acceleration slowing down

```
PROG
VEL         X,10
X           100,Q
END
```

2) point to point movements with change of the velocity profile

speed a

speed b

0 acceleration

FCRAL

```
PROG

VEL         X,10        ; speed a
X           100
WNZ         FCRAL       ; wait slowing down limit switch
VEL         X,5         ; speed b
WEND        X,Q         ; wait quiescent status
END
```

3) interpolated movement

```
PROG

VEL         I,1
ICNT
CIRCLE      XY,OR,6,100,0.5
ECNT
LINEAR      XY,6,0,100
END
```

VIDEO

Message Display on the screen

VIDEO message[,P]

message:	Message text (max 32 alphanum. char.) to be displayed.
P: **P**	Option to require message saving on the disk.

Allows to display, on the PC screen, a direct **message text** and, optionaly, to save them into a disk file.

The characters exceeding the maximum length are automaticaly truncated by the Compiler.

If the VIDEO argument is avoid, the previous displayed message will be erased from the screen.

If the **P** option is used and in the Setup mode the **VIDEO message save** option are been enabled, when the PC receives the message, provides to store them in the REPORT file of the disk.

See also the instruction: MESSAGE.

For instance:

```
        PROG
        VIDEO   SET HYDRRAM          ;message
        SET     PST                  ;set hydrram
        VIDEO   WAITS LIMIT SWITCH   ;message
        WIZ     FCPST,MSGERR         ;waits limit switch
        VIDEO                        ;erase message
        END                          ;end of program

MSGERR: VIDEO   LIMIT SWITCH NOT DETECTED,P
        END
```

VOUT

Set a voltage on the Analog Output line

VOUT dac,voltage

dac: **1÷6**	Analog output number.
voltage: **-10.000÷10.000**	Analog voltage value (in Volt).

Allows to set an analog **voltage** on one of the 6 digital to analog converters (**dac**), numbered as:

1 analog output on the axis: X
2 analog output on the axis: Y
3 analog output on the axis: Z
4 analog output on the axis: W
5 analog output on the axis: V
6 analog output on the auxiliary line (U)

WEND

Wait of an axis or interpolation status

WEND axes,status

WEND ACaxe,status

WEND interpolation,status

axes: **X Y Z W V**	Axes names
status: **A R D F Q**	Status to be waited
ACaxis: **X1÷U3**	ACaxis name.
interpolation: **I**	Selects interpolation.

Waits until the selected **axes** enters in the programmed **status**, before fetching next instruction.

The available status are:

A - acceleration (only for DC axes)
R - steady
D - slowing down
F - end of theorical movement
Q - axis in position (quiescent mode)

When an axis movement is started, by an instruction of (MOV) typology, the axis is placed in the '**A**' status and, at the end of the acceleration ramp, in the '**R**' (steady) status, unless the deplacement length would be less then the ramp space: in this case, the axis enters directly in the '**D**' status, beginning the slowing down ramp. When the theorical quote arrives to the final position, the axis is placed in the '**F**' status, drived only by the residual error of the control loop. From this moment, the system waits until the real axis position is equal to the theorical quote (with the precision defined by the *window* parameter), to put the axis in the '**Q**' status.

If used in the interpolated movements, the available status to test are: **R**,**D**,**F**.

WIZ

Waits if (bitport) is Zero

WIZ bitport

WIZ bitport,label

WIZ bitport,num[,C]

WIZ bitport,num[,W]

bit: **0÷7**	Bit Number
port: **000÷255**	Port address
label:	Label to jump.
num: **1÷255**	Number of Cycle error
C: **C**	If present, the program execution follows after the error detection.
W: **W**	As the previous, but used in special statistics environ

If the programmed (**bit-port**) is zero (0), program waits until it becomes one (1) before fetching next instruction.

The bitport argument may be also defined in mnemonic format, according to the I/OR Definitions file.

If the optional **label** argument is programmed, this instruction triggers a timeout counting (with the value defined in the Module General Setup), at the expiring of which an automatic jump to the label is executed (see the Ex. 1).

If the **num** argument is present, at the timeout expiring, the corresponding Cycle error message is transmitted to the PC for displaying (see the Ex. 2).

If no other option (**C** or **W**) is programmed, when the cycle error occurs, the parallel program stops and may be restarted typing the START key: in this case, execution restarts from the WIZ instruction.

If one of the (**C** or **W**) option is present, when the cycle error occurs, the error message is sended to the PC but the program automaticaly restarts from the WIZ instruction.

See also the RES instruction and the Cycle Error file description (Sect. 4).

Exemples:

1) wait with jump to the label

```
            PROG
            SET         HYDRRAM
            WIZ         FCPIST,ERR1         ; waits limit switch hydrram
            END

ERR1:       VIDEO       LIMIT SWITCH HYDRRAM NOT DETECTED
            END
```

2) wait Cycle Error generating

```
            PROG
            SET         HYDRRAM
            WIZ         FCPIST,25           ; waits limit switch hydrram
            END
```

WNZ

Wait if (bit-port) is Not Zero

WNZ bitport

WNZ bitport,label

WNZ bitport,num[,C]

WNZ bitport,num[,W]

bit: **0÷7**	Bit Number
port: **000÷255**	Port address
label:	Label to jump.
num: **1÷255**	Number of Cycle error
C: **C**	If present, the program execution follows after the error detection.
W: **W**	As the previous, but used in special statistics environ

If the programmed (**bit-port**) is not zero (1), program waits until it becomes zero (0) before fetching next instruction.

The bitport argument may be also defined in mnemonic format, according to the I/OR Definitions file.

If the optional **label** argument is programmed, this instruction triggers a timeout counting (with the value defined in the Module General Setup), at the expiring of which an automatic jump to the label is executed.

If the **num** argument is present, at the timeout expiring, the corresponding Cycle error message is trasmitted to the PC for displying.

If no other option (**C** or **W**) is programmed, when the cycle error occurs the parallel program stops and may be restarted typing the START key: in this case, execution restarts from the WNZ instruction.

If one of the (**C** or **W**) option is present, when the cycle error occurs, the error message is sended to the PC but the program automaticaly restarts from the WNZ instruction.

See also the SET instruction and the Cycle Error file description (Sect. 4).

WPROG

Wait of the one or more Parallel programs or functions end.

WPROG num

WPROG program names

WPROG !num

WPROG functions names

num: **1÷8**	Number of parallel programs
program name:	Parallel program name.
!num: **!0÷255**	Number of parallel function.
function name:	Parallel function name.

Used in multiprogramming mode, allows to wait the end of the parallel programs or function execution before fetching next instruction.

See also the instructions: APROG, EPROG, FPROG, RPROG, SPROG, TPROG.

WSYNC

Wait Synchronisme

WSYNC A

WSYNC num

A: **A**	Waits synchronisme from All the stations
num: **0÷15**	Number of the station which synch. must be waited

Allows the Station 0 (Master) to wait a synchronisme signal from the (**num**) or all (**A**) Slave stations or the one Slave station to wait from the Master.

This instruction assumes different meaning if used by the Master or Slave station:

- **on the Master station**: waits the synchronisme from the selected Slave station; on this last must be present the corresponding SYNC (send synchronisme) instruction.

- **on the Slave station** : waits the synchronisme from the Master (0) station; on this last must be present the corresponding SYNC num (send synchronisme) instruction.

Then, on the station 0, may be programmed:

```
WSYNC   A    or
WSYNC   num
```

on the Slave station:

```
WSYNC   ( 0 assumed by default)  or
WSYNC   0
```

See also the instruction: SYNC.

XOR

2 bytes Exclusive-OR

XOR op1,op2[,result]

op1: **bitport**	Operand 1.
op2: **bitport**	Operand 2.
result: **bitport**	Result.

This instruction executes the bit per bit Exclusive OR enter the (**op1** and **op2**) operand, storing the result in the op2 register or in the **result**, if present.

The **version 4.4E** includes the following new features:

Setup Operating Mode:

- **Parameter: Bactery**

  This parameter is forced to YES, by default.

Automatic Mode:

- **Functions, programs and Tables of Quotes transmissions**

  Returning from Edit by Main Menu, after a program or functions or Tables of quotes modifying, a total Re-transmission is forced.

- **Functions Transmission**

  The Functions transmission is performed only if the END function is ended.