



# CNC

## STD 1

### PROGRAMMING MANUAL FOR S&h CONTROLLERS

- Goya
- Picasso2000
- Rubens

**MANUAL VERSION:** 1.2

<b>Refers to software versions:</b>	CNC	3.90 / 4.12
-------------------------------------	-----	-------------

**MANUAL CODE :** MA CNC U STD1 12  
**DATE:** 13 MARCH 2001



The instructions contained in this manual have been compiled by S&h to be used exclusively by users of S&h controllers.

The contents herein are the property of S&h and may not be reproduced in whole or in part without the written permission of S&h.

The information contained in this manual is subject to change without notice and does not imply any commitment on the part of S&h.

© *Copyright 2001 S&h. All rights reserved.*

## CONTENTS

<b>CHAPTER 1 :</b>	<b>GENERAL OBSERVATIONS</b>
<b>CHAPTER 2 :</b>	<b>INTRODUCTION TO THE PROGRAMMING</b>
<b>CHAPTER 3 :</b>	<b>“IMMEDIATE” INSTRUCTIONS (CONTROL INSTRUCTIONS)</b>
<b>CHAPTER 4 :</b>	<b>“@” e “%” INSTRUCTIONS (CHARACTERISATION INSTRUCTIONS)</b>
<b>CHAPTER 5 :</b>	<b>“G” INSTRUCTIONS (OPERATIVE INSTRUCTIONS)</b>
<b>CHAPTER 6 :</b>	<b>PARAMETER PROGRAMMING</b>
<b>CHAPTER 7 :</b>	<b>“&amp;” INSTRUCTIONS – SPECIAL FUNCTIONS</b>
<b>CHAPTER 8 :</b>	<b>“M” INSTRUCTIONS</b>
<b>CHAPTER 9 :</b>	<b>PROGRAM HANDLING INSTRUCTIONS</b>

### CAUTION

If the system is to behave correctly, it is important that the instructions contained in the manual are followed correctly and, particular attention is paid to the topics marked with the following symbols:



Is associated with facts or circumstances that can cause damage to the system, to the equipment or to the operator.



Is associated with information to take into consideration to prevent damage to equipment in general.



Is associated with operations that must be followed carefully to ensure full success of the application.





# CNC

## **PROGRAMMING MANUAL FOR S&h CONTROLLERS**

### **CHAPTER 1**

-

### **GENERAL FEATURES**



# CONTENTS

Chapter 1 - General features .....	1
<b>INTRODUCTION</b> .....	<b>4</b>
<b>CONFIGURATION</b> .....	<b>4</b>
<b>VERSION</b> .....	<b>4</b>
<b>GENERAL STRUCTURE</b> .....	<b>5</b>
1) CNC.....	5
2) PLC .....	5
3) AUXILIARY PROCESSES.....	5
<b>COMMUNICATION MODES</b> .....	<b>6</b>
<b>POWER-UP</b> .....	<b>7</b>
<b>CNC OPERATIONAL MODES</b> .....	<b>8</b>
<b>HANDLING INSTRUCTIONS IN THE “LINE” MODE</b> .....	<b>9</b>
<b>HANDLING INSTRUCTIONS IN THE “AUTO” MODE</b> .....	<b>10</b>
<b>INTERNAL STRUCTURE OF THE CNC</b> .....	<b>11</b>
"INTERFACE" SECTION .....	11
"CONTROL OF THE AXES " SECTION.....	11

## INTRODUCTION

This manual contains information about the instructions and programming for the CNC section of S&h controllers.

It is designed to be a useful users' guide and quick reference as well as a complete catalogue of all the CNC functions available in the controllers.

The following chapters treat the individual families of instructions, explaining the details of each instruction.

## CONFIGURATION

A few instructions and/or functions of the controllers described in this manual refer to options and are only meaningful if the option referred to has been ordered and supplied for the particular controller.

The instructions that are dependent on installed options are clearly indicated in the manual.

## VERSION

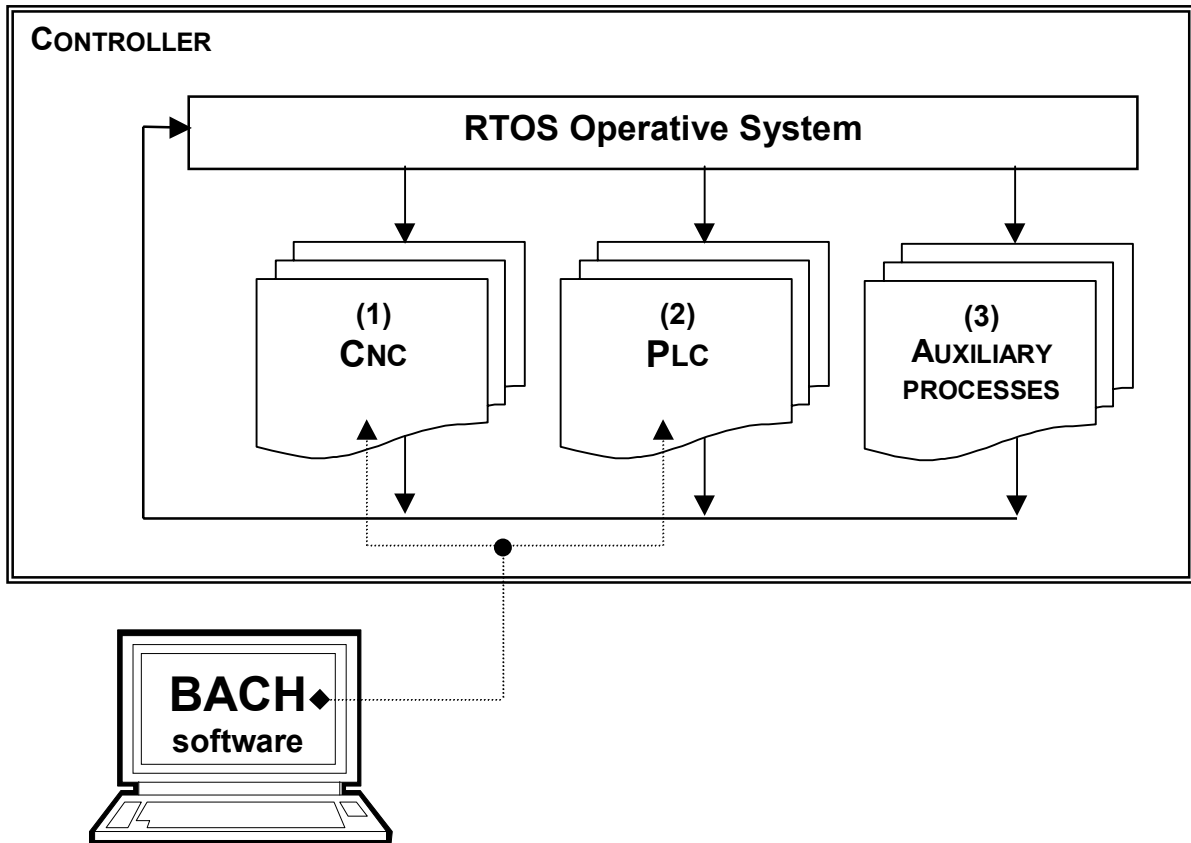
The information in this manual refers to the "Software version of the CNC section of the controllers" indicated on the cover.

Once communication has been established with the controller, the version of software installed can be read using the "**%99**" instruction (look up this instruction in the manual for further details).



## GENERAL STRUCTURE

The internal organisation of S&h controllers is such that the firmware installed in them contains an operative system that can handle several processes in parallel. The structure can be described by the following diagram:



### 1) CNC

This is the process that is concerned with the control of the axes and all the operations related to this function.

### 2) PLC

this is the process that handles the machine I/O independently and autonomously with respect to the Cnc, even though they can be synchronised (the Plc is an option).

### 3) AUXILIARY PROCESSES

These auxiliary services include communications, error detection, timing, alarms, watchdog etc.

#### Note 1:

When the controller is connected to a personal computer over a communications link, it is possible to handle the Cnc(1) and Plc(2) processes directly using the "BACH" support software, which makes available functions such as: upload/download of programs, configuration and calibration of the parameters of the axes, writing, compilation and debugging of Cnc and Plc programs, etc.

## COMMUNICATION MODES

The controller is connect with the outside world by a serial or ISA bus communications link and this allows, using the appropriate protocol, the sending of instructions to and receiving of replies from the controller.

The communication links currently available are the following:

RS232; RS422; ISA\_Bus (Picasso-2000 on the PC)

Refer to the users' manual and installation manual for the electrical connections and hardware specifications (baud rate, data bits, etc.)

Note: Software packages are available for Windows98 that enable an immediate interface with the S&h Cnc controllers. These are:

- **"BACH"**;  
for programming, configuring and handling of operations such as Jog, Automatic, upload/download of programs etc.
- **"TXRXCNC"**  
for updating the firmware in the controller.
- Communications driver **"COMCNC.DLL"**;  
utility for interfacing the user program for handling the communications with the controllers.

These packages automatically handle the communication modes. The user of these packages need not concern himself with the modes of communication employed by the controllers or the procedures used for establishing a colloquium, as described below.

S&h controllers are set up to communicate according to one of the following software protocols:

- 1) terminal mode
- 2) computer without echo mode
- 3) computer with echo mode

### TERMINAL MODE

The main features of this mode of communication are:

The echo of received data is always executed, except for the control codes `<dc1>` (XON), `<dc3>` (XOFF), `<bs>` (^H), `<can>` (^X), `<syn>` (^V).

The echo of the control codes, except for the code `<cr>` is executed by sending the corresponding ASCII lower case character: the echo of the code Ctrl-A = `<SOH>` will therefore be constituted by the character "a".

The echo of the control code `<cr>` is executed sending the code `<lf>` (line feed) followed by the code `<cr>`.

In reply to any instruction, the controller always sends two control codes `<lf>` and `<cr>` before the `<PROMPT>`.

### COMPUTER WITHOUT ECHO MODE

The main feature of this mode of communication is that the echo of data is never executed. The controller only relies to instructions received by sending the requested data, error messages and the `<PROMPT>`.

### COMPUTER WITH ECHO MODE

This is the default communication mode that is selected automatically after every power-up and re-initialisation.

The main feature of computer with echo mode of communication is the fact that the controller always returns the echo of received data, except for the control codes `<dc1>` (XON), `<dc3>` (XOFF), `<bs>` (^H), `<can>` (^X), `<syn>` (^V), `<bel>` (^G).

The echo is simply the return transmission to the PC of the data received, whether it be a character or a control code.

## POWER-UP

On power-up, the controller performs a series of internal tests.

If any errors are detected, the CNC awaits the <CR> code over the communication link. If no errors are detected, the controller starts to execute the internal program No.100 and begins to operate. If the Cnc program No.100 is empty, the CNC awaits the code <cr> over the communication link.

On power-up, when the controller is connected, the communication link is in the OFF state, and the hook-up instruction must be transmitted (see immediate instruction **^G**).

The communication hook-up procedure with the controller is as follows:

1- Transmit the hook up instruction **^G^GC** (ASCII code: 07 07 67)  
 if more than one CNC is connected on the same link (e.g. RS422 in multi-point), it is necessary to first select the controller with which you wish to communicate among those connected **^G^Gnn** and then send the hook-up instruction of the CNC process **'C'**.  
 The code **nn** is the No. of the controller to "hook-up" to over the communications link (see instruction **&12**)

2 – send the control code <cr> (ASCII code 13)

3 – read the reply from the board up to the "PROMPT" character (\$ or ~ or | or\_ )

The string sent by the controller up to the reception of the first <cr> code can be one of two types:

a) if no error has been detected during the initialisation phase of the Cnc

<cr> **#i01** <PROMPT>

No error; the controller is ready to receive any instruction that may be sent from the computer. Note that the code <cr> is nothing more than the echo of the code received.

b) if an error has been detected during the initialisation phase of the Cnc

<cr> **?i-xnn #i01** <PROMPT>

An functional or memory error of the permanent data has been detected. The controller is however ready to receive instructions from the computer.

"**i-xnn**" is a string of ascii characters that identifies the errors encountered (see error messages).

## CNC OPERATIONAL MODES

Four different operational modes are provided for S&h controllers with regard to the CNC processes. These different modes can be selected/activated through instructions sent over the communications link or directly from an external selector switch on the operator's control panel.

The operational modes provided are: LINE, PROG, AUTO and JOG

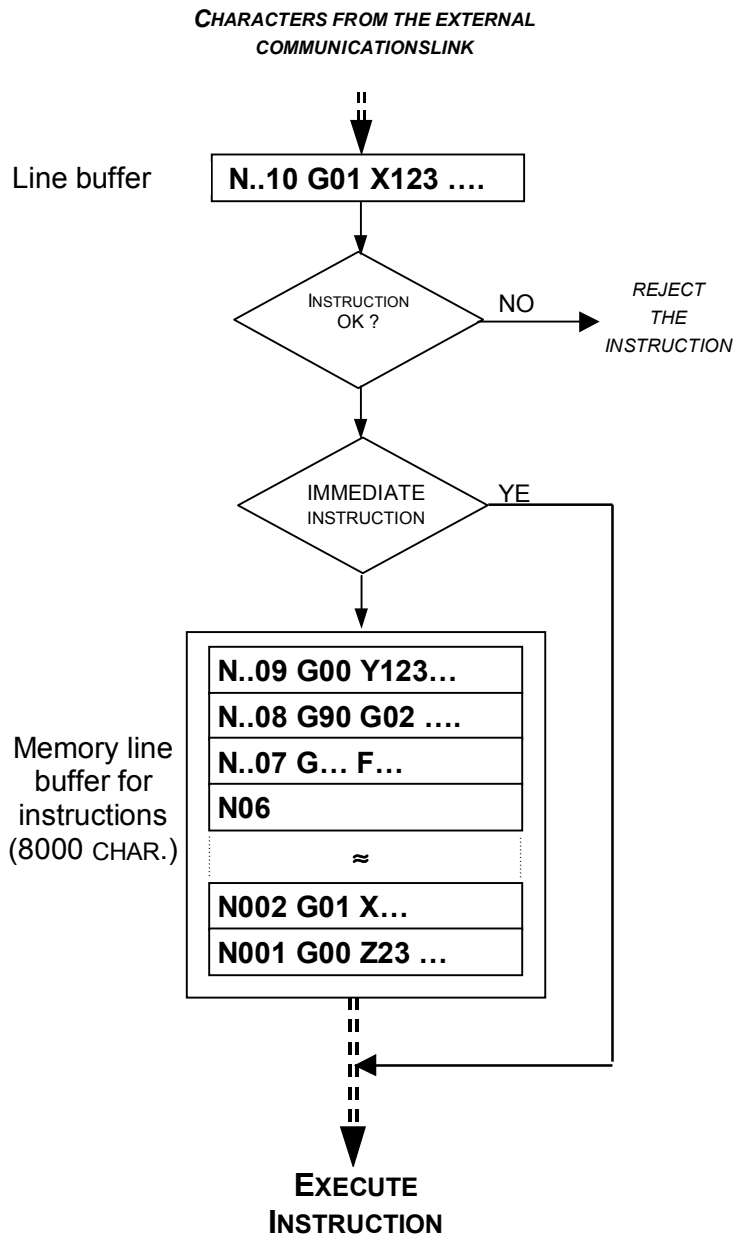
Each mode is identified by a different <PROMPT> sent by the CNC over the communication link in reply to the instructions received from the Personal Computer.

Operational Mode	<Prompt>	Description
<b>LINE</b>	\$	In this mode, the default mode selected automatically every time the Cnc is powered up (if an external operator panel is not fitted), the controller is slaved to the communications link, awaiting instructions to execute. The operational mode ensures that the instructions sent are executed immediately (passing through a FIFO memory that acts as an accumulator).
<b>PROG</b>	~	In this mode, upload to, download from and cancelling of Cnc programs in the controller memory are permitted. The programs are assigned a number from 1 to 99 for user programs and from 100 to 199 for the machine configuration programs reserved for the manufacturer.
<b>AUTO</b>		The AUTO mode is that in which the controller is allowed to execute user programs stored in its memory and that have been previously downloaded in the PROG mode. The programs that may be executed in the AUTO mode are numbered from 1 to 99. They are controlled by START (to initiate), STOP (to stop execution) and ABORT (to cancel the program) instructions.that can be supplied over the communications link or from an operator panel.
<b>JOG</b>	-	In this mode, the JOG (manual) functions can be activated either from the operator control panel or from the appropriate instruction over the communications link. The JOG functions: movement of the axes X+, X-, Y+ etc are customisable and associated with the programs 101 to 136 in the controller's memory.

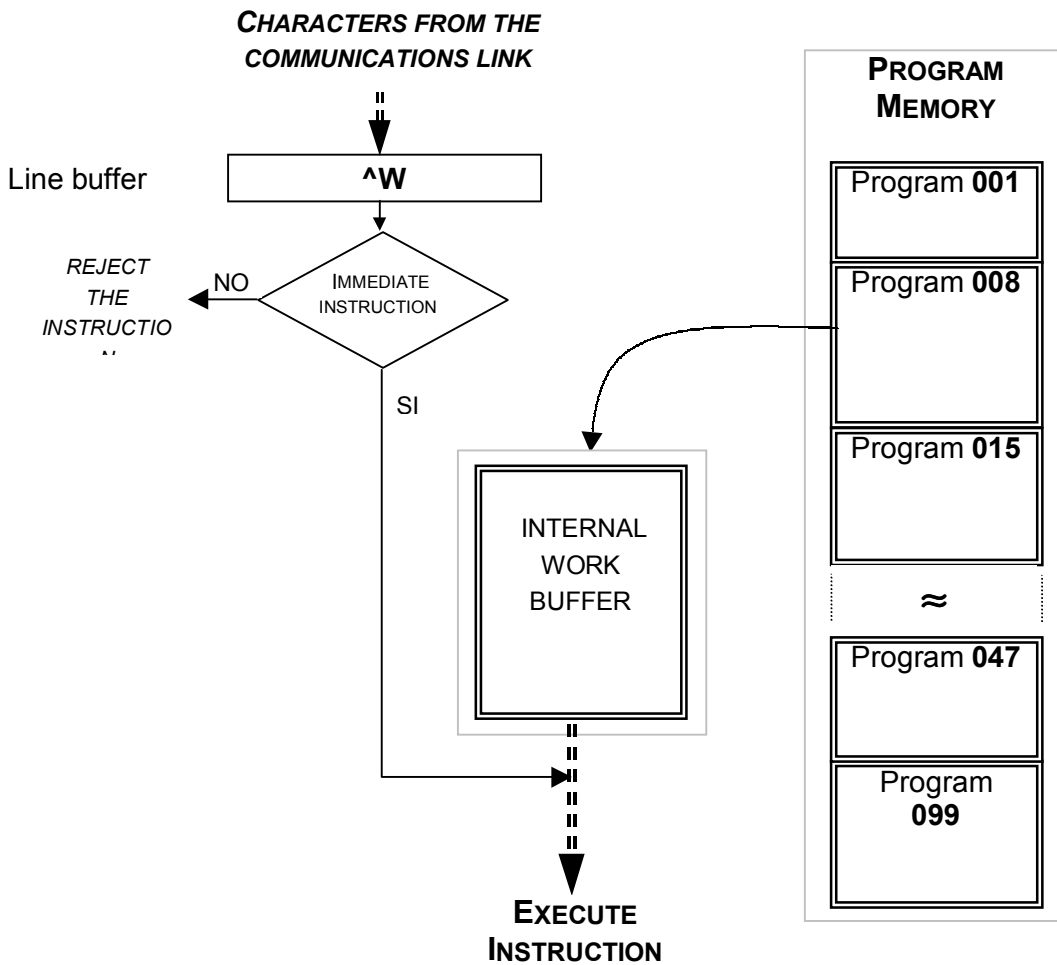
**note:**

In the AUTO and JOG modes, while the program is running, it can still receive instructions from the communications link (immediate execution instructions) requesting data and information on the activity of the Cnc.

# HANDLING INSTRUCTIONS IN THE “LINE” MODE



## HANDLING INSTRUCTIONS IN THE “AUTO” MODE



### PROGRAM MEMORY

The memory, dedicated for work programs, with internal battery back-up so that the data is maintained even when power is removed.

It is also possible to perform all the operations such as uploading (receiving), downloading (transmitting) and cancellation of the Cnc programs from the Personal Computer.

The area of the memory reserved for the PROGRAM MEMORY depends on the controller used.

### INTERNAL WORK BUFFER

Is that area of the memory dedicated to accept the part of the program being executed from the program memory.

## INTERNAL STRUCTURE OF THE CNC

Inside the software that handles the operations associated with the CNC section, two distinct sections of logic can be identified that perform different functions.

The two sections are :

- 1) "INTERFACE"
- 2) "CONTROL OF THE AXES"

### "INTERFACE" SECTION

The "INTERFACE" section performs the following major functions:

- Handles the interface with the outside world
  - handles the external communication links with the outside
  - handles the "operator control panel"
  - receives the instructions, returns the replies and "system prompt"
- Handles the operation modes
  - selects one of the 4 modes of operation (LINE, PROG, AUTO or JOG) that determines the different behaviour of the controller.
- Analysis and treatment of the instructions
  - checks the correctness and admissibility of the received instructions
  - in function of the type of instruction and the active operation mode, performs the instruction or passes it along to the "CONTROL OF THE AXES" section.
- Handles the programs
  - uploads (stores), downloads (executes), cancels, listing of the programs.
- Interfaces with the PLC section
  - data exchange with the PLC section

### "CONTROL OF THE AXES " SECTION

The "CONTROL OF THE AXES" section performs the following major functions:

- Control of the axes
- Execution of instructions

The AXES section then, does not perform any analysis of the instructions, but is uniquely designed for their execution and continuous control of the state of the INSTRUCTIONS FIFO memory, and, if this is not empty, downloads the first available instruction in the working buffer and executes it.







# CNC

## **PROGRAMMING MANUAL FOR S&h CONTROLLERS**

### **Chapter 2**

-

### **Introduction to the Programming**



**CONTENTS**

Chapter 2 - Introduction to the Programming ..... 1

  general ..... 4

    Instructions ..... 4

    Program ..... 5

    Instruction Blocks ..... 5

    Function Blocks ..... 6

    Characterisation Blocks ..... 6

    Assignment Blocks ..... 6

    Comment Blocks ..... 7

    Instructions ..... 7

    Parameters of the instructions ..... 9

    Data Format ..... 10

    Address letters ..... 11

    Programmable functions ..... 13

  Machine registers ..... 21

    Register : condition register ..... 21

    Flag I : conditional jump flag ..... 21

    Machine status register: "D" ..... 22

    Alarm condition register: "E" ..... 23

    Axis dedicated inputs status register : ..... 24

    Axis dedicated outputs status register: ..... 24

  Control panel ..... 25

    Control Panel Register ..... 25

  Error Messages ..... 26

    Error Families ..... 26

    List of Error Messages ..... 27

  Emergencies ..... 28

    Intervention of the controller ..... 28

    Indication ..... 28

    Causes ..... 28

    Emergency reset ..... 28

## GENERAL

The programs executable by the CNC are written using a specific language defined by the ISO standard plus the necessary extensions that are not necessarily supported by this standard. This chapter describes the elements of the language and introduces a few topics regarding the programming and summarises the various instructions.

## INSTRUCTIONS

The instructions are treated in the chapter that describes the specific activities. The description of each instruction is accompanied by the following information:

- name of the instruction
- function
- syntax
- parameters
- characteristics and notes
- examples

Wherever possible, the examples contain parts of a program and diagrams to illustrate the way in which the instructions operate.

### CONVENTIONS OF SYNTAX

In the descriptions of the instructions, the following conventions are used:

- [...] Square brackets indicate optional parts of the instructions
- (n) Round brackets are used to enclose the description of a numerical or alphanumerical field, for example G0 X(absolute co-ordinate). The instruction to position the X-axis at the absolute co-ordinate of 150 will be written **G0 X150**.
- (...) Three dots in round brackets represent fields of parameters that can be present, depending on previous choices made. For example, G68 O(n) represents the syntax  
G68 O1  
G68 O2, "(message)", (r)
- "..." The inverted commas delimit alphanumerical fields and must appear in the instruction. For example **G68 O2, "CNC in emergency", 2**
- <cr> The ascii control characters are closed between the characters < > and highlighted by italics.

**PROGRAM**

A program is made up of a sequence of instructions, written using the characters of the ASCII standard, called **BLOCKS**.

The programs resident on board the CNC are identified using numbers. The permitted range is from 1 to 199.

The dimension and/or quantity of programs that can be stored depends on the memory capacity of the CNC being used (see the installation and interface manual).

**INSTRUCTION BLOCKS**

The maximum length of a block is 128 characters. A block must always end with the ascii code <cr> (carriage return) and can optionally begin with a field defines as the **BLOCK NUMBER**.

The block number allows the programmer to number the blocks of program. It is always made up of the letter “**N**” followed by the numerical value with a maximum of four digits in the range 0 ...9999.

After a block, comments may be inserted that always begin with the character **;** followed by the sequence of alphanumeric characters that define the comment itself.

**GENERAL STRUCTURE OF A BLOCK**

Block :	Number of block	Sequence of characters	Comment	End Code
Example :	N20 [optional]	G01 X123 Z-15.7 F200	; This is a comment [optional]	<cr>
<hr style="width: 100%;"/> Maximum length = 128 characters				

**Note:** In the programming examples in this manual, the code <cr> will be inserted only if necessary for a better comprehension of the example.

**TYPE OF BLOCK**

There are the following types of block:

- function blocks
- characterisation blocks
- assignment blocks
- comment blocks

**ADDRESS**

An *address* is a letter that identifies the type of instruction. For example, the following are addresses:

G, X, Y, Z, F

The ADDRESSES are always UPPERCASE LETTERS

**WORD**

A *word* is an address followed by a numerical value. For example, the following are *words*:

G01 X12.3 Z-15.7 F200

The numerical value assigned to a *word* must be expressed in the measurement system of the *word*.

If the numerical value has a decimal fraction, insert the decimal portion after the decimal point.

ADDRESSES	WORDS
G	G01
X	X12.3
Z	Z-15.7
F	F200

### FUNCTION BLOCKS

These represent the main instructions executable by the controller. In this category fall, for example, the **movement blocks** for the axes.

In general a function block is made up of a series of **programmable functions** or **WORDS**, in turn made up of an upper case letter called the **ADDRESS**, followed by a numerical value.

Block :	Number of block	Word 1	Word 2	Word 3	Word 4	Comment	End code
Example :	N20 [optional]	G01	X123	Z-15.7	F200	; This is a comment [optional]	<cr>
Maximum length = 128 characters							

The block in the example `N20 G01 X123 Z-15.7 F200 ; This is a comment <cr>` represents an instruction for linear interpolation between the axes X and Z (see code G). This contains the number of the block, the comment, the addresses ( G, X, Z, F ) and the words ( G01, X123, Z-15.7, F200 ).

### CHARACTERISATION BLOCKS

The characterisation blocks allow the programming a series of characteristic parameters of the system that enable the function to be adapted to the control of a particular application.

For this reason, they are known as CUSTOMISATION CODES.

In general, a characterisation block is made up of a series of **programmable functions** or **WORDS**, which in turn are made up of an upper case letter, called the **ADDRESS**, followed by a numerical value.

Block :	Number of block	Word 1	Word 2	Word 3	Word 4	Comment	End Code
Example :	N20 [optional]	@60	X1000	Y2000	Z100	; This is a comment [optional]	<cr>
Maximum length = 128 characters							

The block in the example `N20 @60 X1000 Y2000 Z100 ; This is a comment <cr>` represents the definition of the machine limits. It contains the number of the block, the comment, the addresses ( @, X, Y, Z ) and the words ( G01, X1000, Y2000, Z100 ).

### ASSIGNMENT BLOCKS

The assignment blocks enable the programmer to assign a value to the program **variables**. There are the following types of assignment:

Type of assignment	Example
Simple assignment	Q10=123.456
Mathematical expression assignment	Q18=Q7*12+0.5

## COMMENT BLOCKS

These allow the programmer to insert some phrases freely in the program, with the aim of, for example, describing the functions to be performed or to provide elements of information to make the program more easily understood and better documented.

A comment block does not produce messages for the operator. During the execution of the program, the comments are ignored by the controller.

The first character of a comment block must be a semicolon ";" . The rest of the comment block is a sequence of alphanumeric characters, with the exception of the characters: "\$" "~" "|" "-". It should be noted that the four characters used for the *PROMPT* (one for each operational mode) are never accepted in lines of instruction sent to the controller, not even if they are contained in comments (i.e. preceded by the character ";", or in the program "tags" (titles). If they are sent, they are ignored by the controller.

Example:

```
G01 X12.3 Y-45.6 F1000 ;This is a comment <cr>
G02 X52.7 Y27.9 R-174 F500 ;Execute an arc of a circle <cr>
```

## INSTRUCTIONS

In the examples of communications supplied with each instruction, there a "spaces" that can be used to separate the various "words" that make up an instruction and render it more legible well as the control code <cr> that marks the end of the instruction. Note that the "spaces" (ASCII character Hex 20) has significance, if the instructions employ variables.

Therefore, to the controller, the following instructions are identical:

```
'G01 X123 Y456 F1000' <cr> is the same as 'G01X123Y456F1000' < cr >
```

while the following instruction is erroneous due to an incorrect use of the "spaces".

```
'G 01 X 123 Y 456 F 1000' <cr>
```

All the instructions sent to the controller and the relative replies generated are made up of strings composed of standard ASCII codes.

The instructions sent have a minimum of 1 code (see control codes) and a maximum of 128 ASCII codes.

The replies generated in turn can have a minimum of one code (prompt `_$`) and a maximum of 255 ASCII codes.

All the instructions provided in the system can be subdivided into two categories:

- 1 – **execution**
- 2 - **reading**

All the instructions the presuppose the execution of a certain operation by the controller fall into the first category

The controller replies to all these instructions with the *Prompt* character that is dependent on the mode of operation at that moment.

All instructions that request information from the controller belong to the second category. The controller sends the information requested in front of the *Prompt* character.

The instructions accepted by the CNC section can be grouped into the following families:

- **IMMEDIATE** instructions (control codes)
- **@** - % instructions
- **G** instructions
- **VARIABLE handling** instructions
- **&** instructions
- **M** instructions
- **T** instructions
- **PROGRAM handling** instructions

For the descriptions of the individual families and the individual instructions, refer to the relative sections of this manual.



## PARAMETERS OF THE INSTRUCTIONS

All the instructions sent to the controller are handled by the “*interface*” part (see chapter 1) that analyses (checks for admissibility and syntax) and, according to the current operational mode, executes them or pass them on the “*Control of axes*” section.

The parameters that characterise an instruction are the following:

### SOURCE:

Indicates the origin of the instruction.

It can be the SERIAL port, PC bus or the PROGRAM MEMORY

According to the mode of operation, the CNC section handles only one or both of these sources.

LINE and PROG = handles only the serial port

AUTO and JOG = handles both

### EXECUTION:

Indicates the type of execution of the instruction.

It can be IMMEDIATE, QUEUED, FROM PROGRAM

**IMMEDIATE:** These are instructions that are executed by the analysis procedure straight away after the procedure of analysing them for correctness (syntax) and after verifying their admissibility conditions.

**QUEUED:** These are instructions that must be executed in the order in which they have been sent.

The “INTERFACE” section analyses them (syntax) and as a function of the machine configuration (model and type of axes) and of the active mode of operation checks their admissibility:

If the instruction is allowed, it is treated in one of the following ways.

- it is loaded into the FIFO for execution later, both whether it is the job of the “INTERFACE” section or the “CONTROL OF AXES” section (LINE, AUTO, JOG).
- it is saved in the program memory (PROG)

**FROM PROGRAM:**

These are instructions that are not allowed ON-LINE and JOG, but only in PROG and AUTO.

The CNC section analyses them (syntax) and controls their admissibility.

In execution, these instructions in turn can be of different types and consequently treated in different ways by the CNC section

### ADMISSIBILITY:

Admissibility is the sum of the conditions that determine whether an instruction is allowed or not. These conditions are generally tied to the mode of operation, the mode of the (positioner, interpolator, tangential tool) and their configuration (number of axes, if enabled, control panel etc.)

For the instructions for IMMEDIATE execution, the controller (if necessary) executes these, instruction by instruction, (the condition depends on the instruction itself) and this effectively becomes a check of whether they can be executed (performed immediately if allowed).

For QUEUED and FROM PROGRAM commands, the controller performs these after analysing the syntax. It is not a true check on their executability.

The real executability of these instructions is verified only at the moment they are executed.

## DATA FORMAT

### NOTATION

The DECIMAL notation is always used, except in particular circumstances that are explained separately (Variables).

### SIGN

Programming values with sign is made by preceding the number with the “-“ (negative) or “+” (positive) character (optional).

Negative values are always sent to the controller with the “-“ character. Positive values need not contain the “+” sign.

### INTEGER VALUES

The programming INTEGER values varies from instruction to instruction: the allowed limits and consequently the number of digits (numerical characters) change according to the instruction and the address to which the value refers.

INTEGER values are sent from the controller omitting non-significant zeroes.

### REAL VALUES

The programming of real values can be made with a maximum of 15 characters.

The number of decimal digits (after the “.” character) can vary.

The range of values depends on the type of variable to which it is intended to assign the value and can also be limited by the system initialisation parameters.

In any case, if the number at an input exceeds the maximum range, an error condition will be indicated.

Inside the controller, calculations are performed with an accuracy of 15 digits. It should be emphasised that although the accuracy to which the calculations are performed is to 15 significant digits, the system accuracy for the positions of the axes depends on the parameter @50 and/or @51.

## ADDRESS LETTERS

Address	Function	Field	notes
A (41h)	Not used		
B (42h)	Not used		
C (43h)	Not used		
D (44h)	Machine status register		
E (45h)	Alarm register Cam definition variables		
F (46h)	Velocity definition (mm/min, inch/min, revs/min) Hold time for G04 (s)	0 ... MAXREAL 0 ... 9999.99	
G (47h)	Instructions G	0 ... 99	
H (48h)	Address for the instructions G65—G69	1 ...9	
I (49h)	1st Co-ordinate of the circle centre (mm, inch); definition of inputs for @01	+/- MAXREAL	
J (4Ah)	2nd Co-ordinate of the circle centre (mm, inch)	+/- MAXREAL	
K (4Bh)	Definition of panel keys (@70) Screw pitch	+/- MAXREAL	
L (4Ch)	Pointer jump to "label" (# xxxx) Pointer recall subprogram (:xxxx.xx)	1 ... 9999 1 ... 9999.99	
M (4Dh)	Auxiliary functions	0 ... 99	
N (4Eh)	Number of "block"	1 ... 9999	
O (4Fh)	Communication channel for instructions G65-G69		
P (50h)	I/O interface bit with PLC	0 ... 31	
Q (51h)	CNC variables	1 ... 255	
R (52h)	Circle radius (mm, inch)	+/- MAXREAL	
S (53h)	Definition of analogue voltage ratio (revs-velocity /min)	+/- 99999.99	
T (54h)	Selection of tool	0 ... 31	
U (55h)	Not used		
V (56h)	Not used		
W (57h)	Displacement relative to the W-axis (mm, inch, °) Inputs-outputs relative to the W-axis	+/- MAXREAL	
X (58h)	Displacement relative to the X-axis (mm, inch, °) Inputs-outputs relative to the X-axis	+/- MAXREAL	
Y (59h)	Displacement relative to the Y-axis (mm, inch, °) Inputs-outputs relative to the Y-axis	+/- MAXREAL	
Z (5Ah)	Displacement relative to the Z-axis (mm, inch, °) Inputs-outputs relative to the Z-axis	+/- MAXREAL	

Address	Function	Field	notes
@ (40h)	Instructions for writing customisation parameters	0 .... 99	
% (25h)	Instructions for reading customisation parameters Instructions for reading Q variables (%81)	0 .... 99 1...255	
& (26h)	Instructions for "special functions"	0 .... 99	
\$ (24h)	System "Prompt" "On Line"		
~ (7Eh)	System "Prompt" "In Programming"		
(7Ch)	System "Prompt" "In Execution"		
- (5Fh)	System "Prompt" "In Jog"		
!	Request confirmation of instruction (or key)		
?	Identifier of "error message"		
# (23h)	Identifier of "label"	1 .... 9999	
: (3Ah)	Identifier of "Subprogram"	1 .... 9999	
,	Field separator		
" (22h)	Start/end Identifier "Program Tag"	16 char. max	
;	Identifier of beginning of comment		
[ (5Bh)	Opens "Program"		
] (5Dh)	Closes "Program"		
{ (7Bh)	Not used		
} (7Dh)	Not used		
\ (5Ch)	Condition register		
' (27h)	Jump control flag (with G21, 22, 31, 32)		
. (2Eh)	Fractional part separator 1.234 123.4		

## PROGRAMMABLE FUNCTIONS

### CO-ORDINATES OF THE AXES

The co-ordinates of the axes are indicated by the addresses **X,Y,Z,W** and can be programmed in the following range of values: -9999999.999... +9999999.999... [mm or inches]

### R CO-ORDINATE

For movements of circular interpolation (codes G02 e G03) the address **R** represents the **radius** of the circle. This function is also programmable in the range of values -9999999.999... +9999999.999... [mm or inches]

### I J CO-ORDINATES

For circular interpolation, the addresses **I** and **J** define the **co-ordinates of the centre** of the arc of the circle independently of the active plane of interpolation (see G16,G17,G18,G19).

The co-ordinate I specifies l'absciss and J specifies the ordinate.

I and J are also programmable in the range -9999999.999... +9999999.999... [mm or inches]

### K FUNCTION

In a screw, this determines the thread pitch.

It is programmable in the range -9999999.999... +9999999.999... [mm or inches]

### F FUNCTION

For movement blocks, the **F** function defines **velocity of displacements of the axes** which must be programmed in the range  $> 0 \dots < +9999999.999$  [mm or inches] / min

### S FUNCTION

The S function specifies the speed of rotation of the spindle.

The association between the spindle speed and output voltage is programmed by @52.

**S** is programmable in the range - 99999.99...+ 99999.99 [speed revolutions/min]

Symbols:

**a...z** = groups; functions belonging to the same group exclude each other.

**\*** = Functions countersigned with an **\*** are active on power-up and after a software reset.

**1** = Functions countersigned with a (1) are only active in the block in which they appear; all the others are modal, which means that they remain active until they are overwritten by another "G" function of the same group.

**2** = Functions countersigned with a (2) must be programmed alone in a block (other functions cannot exist in the same block).

**L** = executable only the "On Line ( \$ )" mode

**P** = executable only the "In Programming ( ~ )" mode

**E** = executable only the "AUTOMATIC ( | )" mode

**M** = executable only the "JOG (-)" mode

**CONTROL INSTRUCTIONS ( IMMEDIATE INSTRUCTIONS )**

These are standard ASCII codes, from the value 00h (0d) to value 1Fh (31d)

If sent to the controller, they provoke the immediate execution of a determined action (see also chapter 3).

<b>PC Control codes ---&gt; CNC (IMMEDIATE instructions)</b>			
<b>code</b>	<b>refer.</b>	<b>function</b>	<b>notes</b>
^@	<i>nul</i>	Not used	
^A	<i>soh</i>	Request status register	LAJ
^B	<i>stx</i>	Stop Cnc	LAJ
^C	<i>etx</i>	Start Cnc	LAJ
^D	<i>eot</i>	Request Emergency register	LAJ
^E	<i>eng</i>	Request error of following	LAJ
^F	<i>ack</i>	Request number of line being executed	LAJ
^G	<i>bel</i>	Selection / Hook-up communication	LAJ
^H	<i>bs</i>	Cancel last character sent	LAJ
^I	<i>ht</i>	Request active speed	
^J	<i>lf</i>	Not used	LAJ
^K	<i>vt</i>	Set Cnc emergency	LAJ
^L	<i>ff</i>	Reset Cnc emergency	LAJ
^M	<i>cr</i>	Instruction string (block) end code	LAJ
^N	<i>so</i>	Selection active address	LAJ
^O	<i>si</i>	Request dedicated inputs status	LAJ
^P	<i>dle</i>	Annul instruction in execution (and FIFO instructions)	LAJ
^Q	<i>dc1</i>	XON (enable data transmission)	LAJ
^R	<i>dc2</i>	Request to send "alarm code"	LAJ
^S	<i>dc3</i>	XOFF (suspend data transmission)	LAJ
^T	<i>dc4</i>	Single Block execution	AJ
^U	<i>nak</i>	Repetition of last message	LAJ
^V	<i>syn</i>	Request outputs status	LAJ
^W	<i>etb</i>	Request actual position co-ordinates	LAJ
^X	<i>can</i>	Cancel character string sent previously	LAJ
^Y	<i>em</i>	Reset Cnc software (software reset)	LAJ
^Z	<i>sub</i>	Synchronisation code	LAJ
^[	<i>esc</i>	Not used	
^\ (backslash)	<i>fs</i>	Request theoretical co-ordinates	LAJ
^] (closing bracket)	<i>gs</i>	Stop Cnc at end of block	LAJ
^^	<i>rs</i>	Not used	
^_ (underscore)	<i>us</i>	Not used	

<b>Indication Codes CNC ---&gt; PC</b>			
<b>code</b>	<b>refer.</b>	<b>function</b>	<b>notee</b>
^G	<i>bel</i>	Indication of message/alarm to communicate	LAJP
^Z	<i>sub</i>	Synchronisation code PC...CNC	LAJ

**CHARACTERISATION INSTRUCTIONS (@ - % INSTRUCTIONS)**

The '@' functions are codes that enable the programming of the characteristic parameters of the system, that enable the controller to be adapted to the application. The '%' functions are dedicated to the re-reading of the system parameters programmed with the '@' codes (see also chapter 4).

<b>@ - %</b>			
<b>code</b>	<b>address</b>	<b>function</b>	<b>notes</b>
@01-%01	I	Definition and reading of "Programmable" digital inputs	PLJA
@03-%03	XYZW	Dedicated input: definition and reading of "end of travel +"	PLJA
@04-%04	XYZW	Dedicated input: definition and reading of "end of travel -"	PLJA
@05-%05	XYZW	Dedicate input: definition and reading "motor driver fault"	PLJA
@06-%06	XYZW	Dedicated input: definition and reading "machine zero"	PLJA
@15-%15	XYZW	Encoder zero encoder: definition and reading activity (0/1)	PLJA
@16-%16	XYZW	Encoder phases: definition and reading count direction (+/-)	PLJA
@20-%20	XYZW	Definition of cam table parameters	PLJA
@21-%21	XYZW	Definition of cam type and number of repetitions	PLJA
@22-%22	XYZW	Definition of programmable digital outputs	PLJA
@30-%30	I	Define I/O Devices on CAN BUS	PLJA
@31-%31	XYZW	Motor drive direction	PLJA
@32-%32	XYZW	Enabling motor drive	PLJA
@33-%33	XYZW	Voltage range: 0...10v / -10v...+10v	PLJA
@34-%34	XYZW	Voltage polarity: activ.(0/1)=axes mov.(+/-)	PLJA
@35-%35	XYZW	Defining Output Polarity in Frequency	PLJA
@40-%40	XYZW	Servo: emergency limit from servo	PLJA
@41-%41	XYZW	Servo: dead band	PLJA
@42-%42	XYZW	Servo: 2 <sup>nd</sup> value of proportional gain	PLJA
@43-%43	XYZW	Servo: value of integral gain	PLJA
@44-%44	XYZW	Servo: value of derivative gain	PLJA
@45-%45	XYZW	Servo: servo sampling interval	PLJA
@46-%46	XYZW	Servo: feed forward action	PLJA
@47-%47	XYZW	Servo: servo integral limit	PLJA
@48-%48	XYZW	Servo: level between the 1 <sup>st</sup> and 2 <sup>nd</sup> proportional gain	PLJA
@49-%49	XYZW	Servo: 1 <sup>st</sup> value of proportional gain	PLJA
@50-%50	XYZW	Encoder pace/pulse: definition and reading of mm <--> imp ratio	PLJA
@51-%51	XYZW	Space/step: definition and reading of mm <--> step ratio	PLJA
@52-%52	S XYZW I	Definition and reading of velocity - n. rev. <--> voltage ratio	PLJA
@53-%53	XYZW	Maximum encoder frequency /step	PLJA
@54-%54	XYZW	Definition axes velocity limit	PLJA
@56-%56	XYZW	Definition of start/stop frequency or velocity	PLJA
@57-%57	XYZW	Acceleration/deceleration ramp	PLJA
@59-%59	XYZW	Axis in position level	PLJA
@60-%60	XYZW	Machine limits: maximum co-ordinates of the field of working	PLJA
@61-%61	XYZW	Machine limits: minimum co-ordinates of the field of working	PLJA
@70-%70	K	Definition and reading activity of operator panel keys	PLJA
@71-%71	XYZW	Definition of data transmission format from the controller	PLJA
@72-%72	XYZW	Axis type: linear / angular	PLJA
@80-%80		Enable the percentage variation of the axes velocity	PLJA
%81	QEO	Reading variables / origins / camme table	PLJA
@82-%82		Error string: enabled / disabled	PLJA
@83-%83		Enable visual line in execution	PLJA
@84-%84		Enable velocity limitation	PLJA
@85-%85		Bell: Enable/disable "Bell" (every ..sec)	PLJA
@86-%86		Modification of the protocol communication (PLC)	PLJA

<b>@90-%90</b>		Centre error level: defines max. error of the centre co-ordinate	PLJA
<b>@91-%91</b>		Definition level of "reached objective"	PLJA
<b>@94-%94</b>		Definition level of TANGENTIAL tool	PLJA
<b>@95-%95</b>		Definition number of TANGENTIAL cutting tool	PLJA
<b>@96-%96</b>		Definition "times" for raise/lower TANGENTIAL tool	PLJA
<b>@97-%97</b>		Enable operator control panel on power-up	PLJA
<b>@98</b>		Recover the parameters from the permanent memory	L
<b>%98</b>		Reading configuration of the control system	PLJA
<b>@99</b>		Store parameters in the permanent memory	L
<b>%99</b>		Read version of software	PLJA



**OPERATIVE INSTRUCTIONS ( G INSTRUCTIONS )**

The **G** codes program preparatory functions for the work (see the following summary table).  
For a more detailed description, refer to chapter 5.

<b>G</b>				
<b>code</b>	<b>address</b>	<b>function</b>	<b>group</b>	<b>notes</b>
* <b>G00</b>	XYZW	Axes positioning	a	PL 1
<b>G01</b>	XYZW	Linear interpolation	a	PL 1
<b>G02</b>	XYZW IJ R	Clockwise circular interpolation	a	PL 1
<b>G03</b>	XYZW IJ R	Anticlockwise circular interpolation	a	PL 1
<b>G04</b>	F	Hold time at end of block		PL 2
<b>G06</b>	XYZW	Velocity controlled axis	a	PL 2
<b>G16</b>	XYZW	Definition of the plane of circular interpolation		1
* <b>G17</b>		Selection of the XY plane	b	PL 1
<b>G18</b>		Selection of the ZX plane	b	PL 1
<b>G19</b>		Selection of the YZ plane	b	PL 1
<b>G20</b>	L	Unconditioned jump to "label" (#xxxx)		P 2
<b>G21</b>	L	Conditioned jump to "label" (#xxxx) if flag('=)1		P 2
<b>G22</b>	L	Conditioned jump to "label" (#xxxx) if flag('=)0		P 2
<b>G25</b>	XYZW	Working limits field: definition of minima		PL 2
<b>G26</b>	XYZW	Working limits field: definition of maxima		PL 2
* <b>G27</b>	XYZW	Working limits field: cancel limits		PL 2
<b>G30</b>	L	Unconditioned recall of subprogram (:xxxx,xx)		P 2
<b>G31</b>	L	Cond. recall of subprogram (:xxxx,xx) if flag('=)1		P 2
<b>G32</b>	L	Cond. recall of subprogram (:xxxx,xx) if flag('=)0		P 2
* <b>G50</b>	XYZW	Cancel displacement of origin of axes		PL 2
<b>G51</b>	XYZW	Recall origin of axes (set-point)		PL 2
<b>G52</b>	XYZW	Displacement of origin of axes		PL 2
<b>G53</b>	XYZW	Origin of axes at this point		PL 2
<b>G54</b>	XYZW	Origin of axes at this point (Software set-point)		PL 2
<b>G57</b>		"AUTOMATIC LINKAGE" active		PL 1,2
* <b>G58</b>		"AUTOMATIC LINKAGE" inactive		PL 1,2
<b>G61</b>		Precise "IN POSITION" stop active		PL 1,2
* <b>G62</b>		Precise "IN POSITION" stop inactive		PL 1,2
<b>G63</b>		"do not wait for" end of movements		1,2
* <b>G64</b>		"wait for" end of movements		1,2
<b>G65</b>	XYZW PDEOH	Await input / status bit (=0)		PL 1,2
<b>G66</b>	XYZW PDEOH	Await input / status bit (=1)		PL 1,2
<b>G67</b>	XYZW POH	Outputs active (=0)		PL 1,2
<b>G68</b>	XYZW POH	Outputs active (=1)		PL 1,2
<b>G69</b>	XYZW PDEO\	Assign status bit to "flag('=)"		PL 1,2
<b>G70</b>		Quote in "inch"	c	PL 1
* <b>G71</b>		Co-ordinate in "mm"	c	PL 1
<b>G74</b>		Enabling and disabling the co-ordinate reading by laser		PL 1
<b>G80</b>		Enable and disable cam table and engage posn.	d	
<b>G81</b>		Define disengage positions of cams	d	
<b>G82</b>		Automatic definition of cam table	d	
<b>G83</b>		Define cam K factor	d	
<b>G84</b>		Define velocity variation in % cam	d	
* <b>G90</b>		Co-ordinate in "absolute"	e	PL 1
<b>G91</b>		Co-ordinate in "incremental"	e	PL 1
<b>G93</b>		"Tangential tool guide" active		1,2
<b>G94</b>		"Tangential tool guide" inactive		1,2
<b>G98</b>		Velocity handling in positioning and interpolation		PL 1

**SPECIAL INSTRUCTIONS ( & )**

These are particular instructions that the controller handles to execute special functions (see the following table).

For further details, refer to chapter 7.

<b>&amp;</b>		
<b>code</b>	<b>Function</b>	<b>notes</b>
<b>&amp;00</b>	Selection of the Programming mode: <PROG>	
<b>&amp;01</b>	Selection of the Execution mode: <AUTO>	
<b>* &amp;02</b>	Selection of the On Line mode: <ON-LINE>	
<b>&amp;03</b>	Selection of the Manual mode: <JOG>	
<b>&amp;04</b>	List of a Cnc program	
<b>&amp;05</b>	List of Cnc programs in memory	
<b>&amp;06</b>	Cancel Cnc program (1..199)	
<b>&amp;07</b>	Load default parameters	
<b>&amp;08</b>	Select communication mode	
<b>&amp;09</b>	Select "language"	
<b>&amp;10</b>	Enable external control panel	
<b>&amp;11</b>	Disable external control panel	
<b>&amp;12</b>	Define number of the controller	
<b>&amp;15</b>	Modify communications protocol (baud, parity)	
<b>&amp;20</b>	Write variables "Q 1...255"	
<b>&amp;30</b>	Change value of movement velocity "F" (in %)	
<b>&amp;31</b>	Change value of rotation speed "S" (in %)	
<b>&amp;51</b>	Stop recording co-ordinates	
<b>&amp;52</b>	Start recording co-ordinates	
<b>Instruction Codes compatible with immediate instructions</b>		
<b>&amp;70</b>	Stop Cnc immediately ( ^B <stx>)	
<b>&amp;71</b>	Start CNC ( ^C <etx>)	
<b>&amp;79</b>	Set CNC emergency ( ^K <vt> )	
<b>&amp;80</b>	Reset CNC emergency ( ^L <ff> )	
<b>&amp;84</b>	Annul instruction in execution ( ^P <dle>)	
<b>&amp;88</b>	Single block execution ( ^T <dc4>)	

Note:

Functions countersigned by an \* are active on power-up and after a software reset.

**M FUNCTION**

M functions activate different functions of the machine (see the following summary table).  
For more detailed description refer to chapter 8.

Code	function	Notes
<b>M00</b>	Stop program (interrupt execution: as ^])	
<b>M01</b>	Abort programs and subprograms currently being executed	1
<b>M02</b>	End MAIN program	1
<b>M10</b>	Set bit 0(F) PLC (stop CNC until bit is reset)	
<b>M11</b>	Set bit 1(F) PLC (stop CNC until bit is reset)	
<b>M12</b>	Set bit 2(F) PLC (stop CNC until bit is reset)	
<b>M13</b>	Set bit 3(F) PLC (stop CNC until bit is reset)	
<b>M14</b>	Set bit 4(F) PLC (stop CNC until bit is reset)	
<b>M15</b>	Set bit 5(F) PLC (stop CNC until bit is reset)	
<b>M16</b>	Set bit 6(F) PLC (stop CNC until bit is reset)	
<b>M17</b>	Set bit 7(F) PLC (stop CNC until bit is reset)	
<b>M18</b>	Set bit 8(F) PLC (stop CNC until bit is reset)	
<b>M19</b>	Set bit 9(F) PLC (stop CNC until bit is reset)	
<b>M20</b>	Set bit 10(F) PLC (stop CNC until bit is reset)	
<b>M21</b>	Set bit 11(F) PLC (stop CNC until bit is reset)	
<b>M22</b>	Set bit 12(F) PLC (stop CNC until bit is reset)	
<b>M23</b>	Set bit 13(F) PLC (stop CNC until bit is reset)	
<b>M24</b>	Set bit 14(F) PLC (stop CNC until bit is reset)	
<b>M25</b>	Set bit 15(F) PLC (stop CNC until bit is reset)	
<b>M26</b>	Set bit 16(F) PLC (stop CNC until bit is reset)	
<b>M27</b>	Set bit 17(F) PLC (stop CNC until bit is reset)	
<b>M28</b>	Set bit 18(F) PLC (stop CNC until bit is reset)	
<b>M29</b>	Set bit 19(F) PLC (stop CNC until bit is reset)	
<b>M30</b>	Set bit 20(F) PLC (stop CNC until bit is reset)	
<b>M31</b>	Set bit 21(F) PLC (stop CNC until bit is reset)	
<b>M32</b>	Set bit 22(F) PLC (stop CNC until bit is reset)	
<b>M33</b>	Set bit 23(F) PLC (stop CNC until bit is reset)	
<b>M34</b>	Set bit 24(F) PLC (stop CNC until bit is reset)	
<b>M35</b>	Set bit 25(F) PLC (stop CNC until bit is reset)	
<b>M36</b>	Set bit 26(F) PLC (stop CNC until bit is reset)	
<b>M37</b>	Set bit 27(F) PLC (stop CNC until bit is reset)	
<b>M38</b>	Set bit 28(F) PLC (stop CNC until bit is reset)	
<b>M39</b>	Set bit 29(F) PLC (stop CNC until bit is reset)	
<b>M40</b>	Set bit 30(F) PLC (stop CNC until bit is reset)	
<b>M41</b>	Set bit 31(F) PLC (stop CNC until bit is reset)	
<b>M99</b>	End subprogram	1

1 = the instructions have no effect if the controller is in the “on-line” mode.

**N. B.:** if the controller is not used with a PLC, the instructions from **M10** to **M41** have no meaning.

**T FUNCTION**

The T function defines the tool required for the work (see the following summary table). It is programmable in the range 0...31.

code	function
T00	Set bit 32(F) PLC (stop CNC until the bit is reset)
T01	Set bit 33(F) PLC (stop CNC until the bit is reset)
T02	Set bit 34(F) PLC (stop CNC until the bit is reset)
T03	Set bit 35(F) PLC (stop CNC until the bit is reset)
T04	Set bit 36(F) PLC (stop CNC until the bit is reset)
T05	Set bit 37(F) PLC (stop CNC until the bit is reset)
T06	Set bit 38(F) PLC (stop CNC until the bit is reset)
T07	Set bit 39(F) PLC (stop CNC until the bit is reset)
T08	Set bit 40(F) PLC (stop CNC until the bit is reset)
T09	Set bit 41(F) PLC (stop CNC until the bit is reset)
T10	Set bit 42(F) PLC (stop CNC until the bit is reset)
T11	Set bit 43(F) PLC (stop CNC until the bit is reset)
T12	Set bit 44(F) PLC (stop CNC until the bit is reset)
T13	Set bit 45(F) PLC (stop CNC until the bit is reset)
T14	Set bit 46(F) PLC (stop CNC until the bit is reset)
T15	Set bit 47(F) PLC (stop CNC until the bit is reset)
T16	Set bit 48(F) PLC (stop CNC until the bit is reset)
T17	Set bit 49(F) PLC (stop CNC until the bit is reset)
T18	Set bit 50(F) PLC (stop CNC until the bit is reset)
T19	Set bit 51(F) PLC (stop CNC until the bit is reset)
T20	Set bit 52(F) PLC (stop CNC until the bit is reset)
T21	Set bit 53(F) PLC (stop CNC until the bit is reset)
T22	Set bit 54(F) PLC (stop CNC until the bit is reset)
T23	Set bit 55(F) PLC (stop CNC until the bit is reset)
T24	Set bit 56(F) PLC (stop CNC until the bit is reset)
T25	Set bit 57(F) PLC (stop CNC until the bit is reset)
T26	Set bit 58(F) PLC (stop CNC until the bit is reset)
T27	Set bit 59(F) PLC (stop CNC until the bit is reset)
T28	Set bit 60(F) PLC (stop CNC until the bit is reset)
T29	Set bit 61(F) PLC (stop CNC until the bit is reset)
T30	Set bit 62(F) PLC (stop CNC until the bit is reset)
T31	Set bit 63(F) PLC (stop CNC until the bit is reset)

All T instructions are always executed in the immediate mode and have no meaning if the controller is not used with a PLC.

**PROGRAM HANDLING FUNCTIONS**

These are special instructions that control the sending of a program, that permit the insertion of labels and subprograms, and also allow the program to be compiled and stored in the permanent memory of the controller (see also chapter 9).

Program handling instructions		
code	function	notes
[	Beginning of program	
“	Tag of the program	
#	Definition of label	
:	Beginning of subprogram	
]	End of program (compilation and storage)	

## MACHINE REGISTERS

### REGISTER \ : CONDITION REGISTER

The condition register \ is automatically set by operations such as arithmetic functions or on request using a “TST” instruction to be then used by the jump instructions.

Condition register : \		
Bit	Description	Notes
00	....	
01	Overflow (fatal error !)	OV
02	Zero	EQ
03	Negative	NG
04	....	
05	Always = 1	
06	Always = 0	
07	....	

The bit = 1 indicates the “true” condition

### FLAG | : CONDITIONAL JUMP FLAG

The Flag bit | is set on request using a “G69” instruction or the parametric function “SFL” to be then used by the jump instructions.

E.g..

; assigns bit\_EQ of the condition reg. to flag | :  
G69 |2

; assigns input X1 to flag | :  
G69 X1

; assigns to flag | the result of the expression :  
SFL((QIX.AND.1).OR.(QRD.AND.27))

**MACHINE STATUS REGISTER: "D"**

The "Machine Status" register enables the system to recognise the condition of the machine at any moment. To the relative request (immediate instruction **^A**) the "D" status register is transmitted in the format **D...X...Y...Z...W...**

The following explains the significance of each individual bit.

Cnc status register CNC : <b>D</b>			
Bit	Description		Notes
00	Operator panel "ACTIVE"	ACTIVE IF = 1	
01	Probe		
02	PLC running		
03	PLC in error		
04	Objective below the setting(@91) in objective zone		
05	Program "IN EXECUTION"		
06	CNC "IN STOP"		
07	CNC "IN EMERGENCY"		

AXES status register : <b>X, Y, Z, W</b>			
Bit	Description		
00	AXIS IN LOCAL ORIGIN	LASER NOT CONNECTED, if G74 = 1, 2 or 3 (see G74)	ACTIVE IF = 1
01	AXIS "IN CAM"	LASER FAULT, if G74 = 1, always 0 if G74 = 2 or 3	
02	Axis "IN STOP"	VALUE BELOW THE MINIMUM, if G74 = 1, always 0 if G74 = 2 or 3	
03	Axis "IN POSITION"	VALUE ABOVE THE MAXIMUM, if G74 = 1, always 0 if G74 = 2 or 3	
04	Axis "IN SATURATION"		
05	Axis set-point EXECUTED		
06	The axis is moving in the POSITIVE (+) direction		
07	Axis IN MOVEMENT		

**ALARM CONDITION REGISTER: "E"**

The "Alarm" register allows the system to recognise the condition of the alarms of the machine at any moment. To the appropriate request (immediate instruction **^D**) the status register "E" transmits in the format **E...X...Y...Z...W...**

The following explains the significance of each individual bit.

CNC alarm register : <b>E</b>			
Bit	Description		notes
00		ACTIVE IF = 1	
01			
02			
03	Emergency activated by probe		
04	Emergency activated by error of execution		
05	Emergency activated by axes control		
06	Emergency activated by emergency input		
07	Emergency activated by immediate instruction <b>^K</b>		

AXIS alarm register : <b>X, Y, Z, W</b>			
Bit	Description		notes
00		ACTIVE IF = 1	
01	Alarm active from axis emergency input		
02	Axis in emergency for motor drive fault		
03	Axis in emergency from limit switch SFW (-)		
04	Axis in emergency from limit switch SFW (+)		
05	Axis in emergency from limit switch HDW (-)		
06	Axis in emergency from limit switch HDW (+)		
07	Axis in emergency from servo control		

**AXIS DEDICATED INPUTS STATUS REGISTER :**

The “Input Status” and “Output Status” registers allow the system to recognise the conditions of each dedicated input and output at any moment.  
The following shows the significance of the individual bits.

VARIABLE **QIX, QIY, QIZ, QIW** : AXIS INPUTS IMAGE “X”, “Y”, “Z”, “W”

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

DEDICATED AXIS INPUTS : **X, Y, Z, W**

Bit	Description		notes
00	Not used	1 = CLOSED  0 = OPEN	
01	Not used		
02	Not used		
03	Not used		
04	Limit switch (-)		
05	Limit switch (+)		
06	Motor drive OK		
07	Machine zero		

**AXIS DEDICATED OUTPUTS STATUS REGISTER:**

VARIABLE **QOX, QOY, QOZ, QOW** : AXIS OUTPUTS IMAGE “X”, “Y”, “Z”, “W”

Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

DEDICATED AXIS OUTPUTS : **X, Y, Z, W**

Bit	Description		notes
00	Not used	1 = ACTIVE  0 = INACTIVE	
01	Not used		
02	Not used		
03	Not used		
04	Not used		
05	Not used		
06	Not used		
07	Not used		



## CONTROL PANEL

The Cnc automatically handles a machine CONTROL PANEL for carrying out all the machine control operations that must be controlled by the operator through easy-to-use manual commands (keys and switches) (e.g. keys for manual movements, Start, Stop, selecting work program, etc.)

Since each key is associated with programs in the Cnc, it is possible to customise the operations of the Control Panel to meet specific requirements.

The Cnc Control Panel handles:

- 16 Function keys
- 1 Mode selection switch
- 1 Program number selector

## CONTROL PANEL REGISTER

The CONTROL PANEL is handled via a 32-bit logic image to which its particular operation is associated. A program is associated with every key both on when it is pressed and when it is released.

The CNC "sees" the control panel in variable PLC 14 (positions 224..255).

VARIABLE QPN : OPERATOR PANEL IMAGE 31...0

Bit 31|30|29|28|27|26|25|24|23|22|21|20|19|18|17|16|15|14|13|12|11|10|09|08|07|06|05|04|03|02|01|00

Bit	Description		
00	Input 00	Key 01	KEYS 1...8
01	" " 01	" 02	
02	" " 02	" 03	
03	" " 03	" 04	
04	" " 04	" 05	
05	" " 05	" 06	
06	" " 06	" 07	
07	" " 07	" 08	
08	Input 08	Key 09	KEYS 9...16
09	" " 09	" 10	
10	" " 10	" 11	
11	" " 11	" 12	
12	" " 12	" 13	
13	" " 13	" 14	
14	" " 14	" 15	
15	" " 15	" 16	
16			
17			
18			
19			
20			
21			
22	Mode selection:	Data 0	MODE SELECTOR
23	" " "	" 1	
24	Cnc program selector (units)	Data 0	PROGRAM SELECTOR
25	" " " "	" 1	
26	" " " "	" 2	
27	" " " "	" 3	
28	Cnc program selector (tens)	Data 4	
29	" " " "	" 5	
30	" " " "	" 6	
31	" " " "	" 7	

\* **Note:** see the following table

Bit 22	Bit 23	Modality
0	0	LINE
0	1	AUTO
1	0	PROG
1	1	JOG

## ERROR MESSAGES

Independently of the type of error, and therefore from the cause that has determined it, an error message is always made up of a string of ASCC characters of the type:

**?0-(m)(nn): (description) where:**

**?0-** ERROR MESSAGE IDENTIFIER: the three initial characters of the string are always present in any error message.

**m** FAMILY CODE: is only one character that specifies the family of errors and is always a **lower case letter**.

**nn** ERROR CODE: uses two numeric characters to indicate the specific error inside the family.

*description* is a string of ASCII characters that describes the error.

Finally, remember that the controller, after the error message, sends the PROMPT, or the character that indicates its current status and, at the same time, signals to the host that it is ready to receive the next instruction.

## ERROR FAMILIES

### CODE DESCRIPTION

- a** message sent following a the execution of a **G68 O2** instruction (the description of the instruction is found in Chapter 5).
- b** errors caused by sending the board a line of instruction that is too long (more than 128 characters) or by sending an incorrect control code.
- c** errors due to an erroneous line of instruction or an instruction that is not allowed.
- e** errors generated during the execution of the instructions: they block the execution of the instruction and it is signalled by sending a periodic <BEL> code.
- f** fatal errors from the PLC.
- g** errors generated by the saturation of the execution buffers in which the sent instructions are stored or of the program memory.
- i** errors due to malfunctions that could occur on power-up.
- k** message sent after the execution of a **G65 O2** instruction (the description of the instruction is found in Chapter 5).
- p** error generated during the compilation of a program.
- w** warning from the PLC.

<b>LIST OF ERROR MESSAGES</b>		
m	nn	Meaning
<b>a</b>	s	Executing instruction <b>G68 O2</b> . s is the parameter indicated in the instruction
<b>b</b>	<b>01</b>	Line buffer full
<b>c</b>	<b>01</b>	Error for the type of parameter passed
	<b>02</b>	Program not found in memory
	<b>04</b>	Decimals not expected
	<b>05</b>	Error for exceeding limits
	<b>06</b>	Instruction not allowed
	<b>08</b>	Unknown instruction, not implemented or not active
	<b>09</b>	Conversion error in the number passed
	<b>10</b>	Error in the programming phase
<b>e</b>	<b>11</b>	Wrong plane selected
	<b>01</b>	Internal error
	<b>02</b>	PLC parameter error
	<b>03</b>	CNC parameter error
	<b>04</b>	CNC in emergency
	<b>05</b>	Host instruction error
	<b>06</b>	Host overrun
	<b>07</b>	INT-C time error
	<b>08</b>	RX error with Host
	<b>09</b>	Overflow of parameters from Host
	<b>10</b>	Division by Zero
	<b>11</b>	Module programming error
	<b>12</b>	Emergency from an immediate instruction
	<b>16</b>	Error in dialogue with modules
<b>17</b>	PLC program loop	
<b>18</b>	Emergency from ^K	
<b>19</b>	Transmission FIFO full	
<b>20</b>	Subprogram stack error	
<b>21</b>	Mathematical calculation error	
<b>f</b>	l	Warning from PLC. l is the code associated with the line of the PLC program that has caused the notification
<b>i</b>	<b>01</b>	E <sup>2</sup> prom checksum
	<b>04</b>	Axes' parameters buffer error
	<b>05</b>	CNC program buffer error
	<b>06</b>	Internal dialogue error
	<b>07</b>	PLC program buffer error
	<b>08</b>	CAN-BUS initialisation error
<b>g</b>	<b>01</b>	Instructions FIFO full
	<b>02</b>	Program RAM saturated
	<b>03</b>	Program handling error
<b>k</b>	s	Executing instruction <b>G68 O2</b> . s is the parameter indicated in the instruction
<b>p</b>	<b>01</b>	Jump control error (G20-G21-G22)
	<b>02</b>	Subprogram control error (G30-G31-G32)
	<b>03</b>	Internal stack error due to excessive nesting
<b>w</b>	l	Fatal PLC program error. l is the code associated with the line of the PLC program that has caused the notification

## EMERGENCIES

### INTERVENTION OF THE CONTROLLER

The emergency condition is handled by the following interventions:

- 1) The execution of any current instruction or program is cancelled.
- 2) Eventual movements being executed are interrupted instantaneously without even the deceleration ramp: for d.c. axes, the analogue output will be set to 0 Volt while, for stepper motor axes, no further pulses will be sent to the motor.
- 3) The ENABLE MOTOR DRIVE output for each axis will be disabled.

### INDICATION

The state of emergency of the controller is indicated by:

- 1) Indication on the display or LED if present.
- 2) Sending (if enabled) the <BEL> code on the communication link (see instruction @85).
- 3) Activation of the (CNC) auxiliary outputs, if programmed, with emergency functions (see instruction @22).

### CAUSES

The causes of the emergency can be many (immediate instruction, activation of an input, servocontrol, etc., etc.). See the immediate instruction Ctrl-D that allows the emergency register to be read.

### EMERGENCY RESET

The state of emergency of the controller can be annulled with the immediate instruction Ctrl-L, activating a programmed input as an emergency reset input or pressing a pushbutton programmed as the reset in a mode where the remote panel is active.

After an emergency reset, it is probable that the co-ordinates of the stepper axes (without feedback) are no longer correct, since, as the deceleration ramp was not followed, the motor could have lost some steps. Therefore, for these axes, it is recommended that the procedure for seeking the machine zero is followed (if foreseen for the application). See instruction G51.



# CNC

## **PROGRAMMING MANUAL FOR S&h CONTROLLERS**

### **Chapter 3**

-

### **“Immediate” Instructions (Control Instructions)**



**CONTENTS**

Chapter 3 - "Immediate" Instructions (Control Instructions).....	3-1
Introduction .....	3-4
Communication control instructions.....	3-4
Cnc control instructions.....	3-4
CTRL_A <soh> Request the Status Register.....	3-5
CTRL_B <stx> Stop Cnc .....	3-6
CTRL_C <etx> Start Cnc.....	3-6
CTRL_D <eot> Request Emergency Registers .....	3-7
CTRL_E <enq> Request Follow Error .....	3-8
CTRL_F <ack> Request No. of Line in Execution.....	3-9
CTRL_G <bel> Hook-up Communication .....	3-10
CTRL_H <bs> Cancel Last Character Sent.....	3-11
CTRL_I <ht> Request Active Velocity.....	3-11
CTRL_K <vt> Set Cnc Emergency .....	3-12
CTRL_L <ff> Reset Cnc Emergency .....	3-12
CTRL_M <cr> Close Line of Instruction.....	3-12
CTRL_N <so> Select Addresses .....	3-13
CTRL_O <si> Request Status of Dedicated Inputs.....	3-14
CTRL_P <dle> Annul Instruction in Execution .....	3-15
CTRL_Q <dc1> Xon (enable data transmission) .....	3-16
CTRL_S <dc3> Xoff (suspend data transmission).....	3-16
CTRL_R <dc2> Request Alarm Code.....	3-17
CTRL_T <dc4> Single-Block Execution.....	3-18
CTRL_U <nak> Repeat Last Message .....	3-18
CTRL_V <syn> Request Dedicated Outputs Status .....	3-19
CTRL_W <etb> Request Actual Co-ordinates .....	3-20
CTRL_X <can> Cancel String Sent .....	3-21
CTRL_Y <em> Reset Cnc (software) .....	3-22
CTRL_Z <sub> Synchronisation Code .....	3-23
CTRL_ \ <fs> Request Theoretical Co-ordinates .....	3-24
CTRL_] <gs> Stop Cnc at End of Block.....	3-25

## Introduction

The codes use standard ASCII, from value **00h** (0d) to value **1Fh** (31d).

If sent to the controller, they cause the execution of a particular action. In other words, they are considered as commands or instructions for **immediate execution**.

In function of their meaning, they can be divided into two categories:

- COMMUNICATION CONTROL instructions
- CNC CONTROL instructions

## Communication control instructions

These control the communications with the equipment and the sending of any line of instruction. They can be sent in any position within the line.

## Cnc control instructions

Some of them cause the execution of a determined action (e.g.: <EM> = software reset), others, on the other hand, request information from the controller (e.g.: <ETB> = read co-ordinates). In the first case, the controller executes the instruction and replies only with the PROMPT for the active operation mode. In the second, the controller sends the information requested followed by the PROMPT.

Normally an immediate instruction is made up of only one control code.

Nevertheless, some of them are exceptions to this rule, inasmuch as they require a further character before being executed.

### Notes

The description that follows only concerns accepted control codes. The eventual sending of an unforeseen control code will be indicated by an error message.

If not specified differently, the communication examples supplied always relate to the “computer without echo” type of communication and the ON-LINE mode of operation

The addresses of the axes Y, Z and W described in the instructions that follow are only allowed if they are available on the model of the controller used.



## CTRL\_A <soh> Request the Status Register

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

### Properties

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

### Description

Immediate instruction **^A = <SOH> = 01h (01d)      REQUEST STATUS REGISTER**

The controller continually updates the 8-bit registers that contain the information concerning its own (CNC) status and the status of the axes handled by the board (XYZW) and that can be requested at any moment using the code <SOH>.

The reply from the board will contain numerical data in decimal format that corresponds to the values of the status register, preceded by a character that identifies the register:

'D' : reg. CNC      'X' : reg. X-axis      'Y' : reg. Y-axis      'Z' : reg. Z-axis      'W': reg. W-axis

The significance of the individual bits in the status register is as follows:

CNC status register : <b>D</b>
--------------------------------

Bit	Description		notes
00	Operator panel "ACTIVE"	ACTIVE IF = 1	
01	Probe enabled		
02	PLC running		
03	PLC in error		
04	Objective under setting (@91)      in objective zone		
05	Program "IN EXECUTION"		
06	CNC "IN STOP"		
07	CNC "IN EMERGENCY"		

AXES status register : <b>X, Y, Z, W</b>
--

Bit	Description		
00	AXIS IN LOCAL ORIGIN	LASER NOT CONNECTED, if G74 = 1, 2 or 3 (see G74)	ACTIVE IF = 1
01	AXIS "IN CAM"	LASER FAULT, if G74 = 1, always 0 if G74 = 2 or 3	
02	Axis "IN STOP"	VALUE BELOW THE MINIMUM, if G74 = 1, always 0 if G74 = 2 or 3	
03	Axis "IN POSITION"	VALUE ABOVE THE MAXIMUM, if G74 = 1, always 0 if G74 = 2 or 3	
04	Axis "IN SATURATION"		
05	Axis set-point EXECUTED		
06	The axis is moving in the POSITIVE (+) direction		
07	Axis IN MOVEMENT		

E.g.: request status register

Versions with 3 axes:

<SOH>      ; request to read the status registers  
 D32X160Y224Z32      ; reply from the controller

Versions with 2 axes:

<SOH>      ; request to read the status registers  
 D32X160Y224      ; reply from the controller

**CTRL\_B <stx> Stop Cnc**

**CTRL\_C <etx> Start Cnc**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^B = <STX> = 02h (02d) : STOP CNC**

Using the code <STX> (^B), it is possible to stop the axes in movement, with a controlled “braking”, as it triggers the deceleration ramp defined in the customisation parameters. The stop condition is indicated by the relative bit in the status registers.

The movement in execution could be concluded subsequently by sending the code for “start” <ETX> (^C). Note that the instruction can be sent even when the axis is stopped: in which case, subsequent movements sent to the board will not be executed until a new <ETX> instruction is sent.

Note: The equivalent instruction to the “immediate instruction **^B**”, but which works in the “PROGRAMMING” mode, is the instruction “**&70**”

Immediate instruction **^C = <ETX> = 03h (03d) : START CNC**

The code <ETX> (^C) can be used to re-start a movement that has been stopped with the “stop” code <STX> (^B).

The instruction is also used to begin the execution of a program in AUTOMATIC whenever the external panel is disabled.

Note: The equivalent instruction to the “immediate instruction **^C**”, but which works in the “PROGRAMMING” mode, is the instruction “**&71**”

E.g.: Stop Cnc

Stop a programmed movement:

```
G00 X1000 <CR>           ; active movement of X-axis to position 1000
<STX>                   ; Stop current movement of X-axis
```

E.g.: Stop / Start Cnc :

Stop axes, load a series of lines of movements, Start movement of axes

```
<STX>                   ; Stop movements
G0 X1000 <CR>           ; activate movement of X-axis to position 1000 (the axis remains
                        STOPPED)
G0 Y1500 <CR>           ; the movement of the Y-axis queued to that of the X-axis (the axes remain
                        STOPPED)
<ETX>                   ; Start movements
```



<b>CTRL_E &lt;enq&gt; Request Follow Error</b>
--

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^E = <ENQ> = 05h (05d) : REQUEST FOLLOWING ERROR**

The instruction <ENQ> (^E) is used to request the controller to send the values regarding the errors of following the axes, even when the axes are in movement.

The values sent will conform to the program executed using the instructions @71 e G70-G71 to define the number of decimal digits and the units of measurement.

The data regarding the axes will be sent conforming to those defined with the instruction **^N** for the selection of the axes to transmit.

The corresponding variables that contain transmitted data are : **QEX, QEY, QEZ, QEW**

E.g.: request following error

<code>&lt;ENQ&gt;</code>	<code>;request to read value of following error</code>
<code>X0.01Y0.57Z1.5</code>	<code>;reply from controller</code>

**CTRL\_F <ack> Request No. of Line in Execution**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^F = <ACK> = 06h (06d) : REQUEST NO. OF LINE IN EXECUTION**

The code <ACK> (^F) can be used to request the number of the line of instructions that the CNC controller is executing at any moment.

The reply will contain a numerical value preceded by the character “N”.

Note: if the parameter @83 = 1, has been programmed in the automatic mode, then, as well as the line number, the string of the line instruction will be sent.

E.g.: request no. of line in execution

```
N10 G01 X100 Y-200 <CR>
N20 X150 Y0 <CR> ←-----
N30 X0 Y100 <CR>
<ACK>
N20 (at this moment this line is being executed ) ●
```

```
N10 G01 X100 Y-200 <CR>
X150 Y0 <CR> ←-----
X0 Y100 <CR>
<ACK>
N11 (at this moment this line is being executed ) ●
```

## CTRL\_G <bel> Hook-up Communication

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^G = <BEL> = 07h (07d) : HOOK-UP COMMUNICATION**

The instruction <BEL> (^G) is special, as it allows:

- Activation of the communications with one of the controllers that are on-line (multi-point communication in RS422)
- Activation of the communications with one of the internal processes to the CNC.

The instruction <BEL> (^G) must be sent twice, since the first <BEL> code is that which frees the communication link (all the controllers/processes close down), the second <BEL> code is that of “attention” that makes the controller prepare itself to receive subsequent characters of operational selection.

To select the “process” with which it is wished to communicate inside the controller, the code **^G** must be made, followed by the letter that indicates the “process” with which it is wished to link the communication:

"O"	=	"OFF"	deactivate the communication (no process linked)
"C"	=	"CNC"	activate the communication with the CNC section
"P"	=	"PLC"	activate the communication with the PLC section
"T"	=	"TEST"	test procedure ( <u>RESERVED</u> )

In the case of communication with more than one controller (multi-point handling with RS422 serial link) use this instruction to hook-up the controller with which it is wished to communicate.

To select the number of the controller “**NN**”, the immediate instruction **^G** must be followed by the number “**NN**” (always 2 digits).

This instruction does not have echo and does not have the “prompt” in reply.

This instruction must always be followed by a hook-up instruction to the “process” (CNC [**^GC**] or PLC [**^GP**] ) of the active controller.

If the controller has the number “**00**”, it is not necessary to send the instruction for selecting the number. To assign a number to a controller, refer to the description of the instruction &12.

E.g.: hook-up of a Cnc with number 05 associated :

```

<BEL><BEL>05           ;select controller no. 5
<BEL><BEL>C             ;activate the communication with the Cnc
<CR>                   ;send a “close line” code
    
```

**Note**

The reception of only one **^G <BEL>** character drops the communication  
 In reply to the instruction **^G <BEL>** **the "PROMPT" is never sent.**  
 For the <BEL> code **the “echo” is never executed**

## CTRL\_H <bs> Cancel Last Character Sent

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	-	

**Description**

Immediate instruction **^H = <BS> = 08h (08d) : CANCEL LAST CHARACTER SENT**

The code <BS> (^H = back space) allows the last character sent to be cancelled. It is obvious that the line of instruction must not have been closed with the code <CR>.

If the code <BS> is sent as the first piece of data in a line of instruction, it has no effect.

E.g.: Correction of a character in a line of instruction:

G01 <BS> 0 X100 <CR>

The instruction sent in the example, after the correction, is : G00 X100<CR>

## CTRL\_I <ht> Request Active Velocity

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^I = <HT> = 09h (09d) : REQUEST ACTIVE VELOCITY**

With the instruction <HT> (^I), it is possible to request the controller to send the values regarding the velocity of the axes (Xvel, Yvel, ..) and that of the “interpolated point” (Fvel).

The values sent by the controller will conform to the way it has been set-up with the @71 and G70-G71 instructions for defining the number of decimal digits and the unit of measurement.

The data regarding the axes will be sent conforming to those defined with the instruction **^N** for selecting the axes to transmit.

The corresponding variables that contain the transmitted data are :

- QPT** (interpolated point velocity)
- QVX, QVY, QVZ, QVW** (axes velocity)

E.g.: request active velocity

```
<ENQ>                ;request to send active velocity
X100Y100 F141         ;reply from the controller
```

**CTRL\_K <vt> Set Cnc Emergency**

**CTRL\_L <ff> Reset Cnc Emergency**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^K = <VT> = 0Bh (11d) : SET CNC EMERGENCY**

With the instruction <VT> (^K) the state of emergency of the controller can be generated at any moment. Any eventual movements taking place will be stopped instantly, without following any programmed deceleration ramp.

On receiving it, the controller updates the dedicated bits in the status and alarm registers.

Note: The equivalent instruction to the “immediate instruction **^K**”, but which works in the “PROGRAMMING” mode is the instruction “**&79**”

Immediate instruction **^L = <FF> = 0Ch (12d) : RESET CNC EMERGENCY**

With the instruction <FF> (^L), the state of emergency can be annulled at any moment, whatever the cause that has provoked it.

On receiving it, the controller updates the dedicated bits in the status and alarm registers.

Note: The equivalent instruction to the “immediate instruction **^L**”, but which works in the “PROGRAMMING” mode is the instruction “**&80**”



During the state of emergency, whatever the cause that has provoked it, the controller disables the motor drives (provided that the relative signal of “enable motor drive” has been programmed as active and the controller output has been wired to the “enable motor drive” input), switching the status of the dedicated outputs dedicated to their consensus.

When the state of emergency is terminated, the controller re-enables the motor drives, re-enabling the output states dedicated to their consensus.

**CTRL\_M <cr> Close Line of Instruction**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	Cannot be sent if an immediate instruction requiring a further character is in execution	

**Description**

Immediate instruction **^M = <CR> = 0Dh (13d) : CLOSE LINE OF INSTRUCTION (END OF STRING)**

The code <CR> (^M = carriage return) is used as an “end of string” code (closes line of instruction). On receiving it, the controller analyses the instructions sent and, if correct, executes them.

Note that the code <CR> can be sent even as the first piece of data in a line of instruction. In this case, the controller replies by sending the PROMPT of the active mode of operation.



## CTRL\_N <so> Select Addresses

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

### Properties

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

### Description

Immediate instruction **^N = <SO> = 0Eh (14d) : SELECT ADDRESSES**

The code <SO> (^N) is a particular immediate instruction that requires a further character to be sent before it can be executed.

The character can be one of the following:

'C'	select CNC (select all the axes)
'X'	select X-axis
'Y'	select Y-axis
'Z'	select Z-axis
'W'	select W-axis

This character determines, from that moment, a different format for the replies to the following immediate instructions:

^E <ENQ>	read following error
^O <SI>	read input states
^V <SYN>	read output states
^W <ETB>	read actual co-ordinates
^\ <FS>	read theoretical co-ordinates
^I <HT>	read velocity

In other words, the addresses present in the immediate instructions are selected.



On power-up and after a software reset, the controller automatically selects the address 'C'(CNC).

E.g.: select active addresses (system with 3 axes X,Y,Z)

```

<ETB>           ;request actual co-ordinates
X100Y200Z0      ;reply from the controller
<SO>X           ;select X-axis
<ETB>           ;request actual co-ordinates
X100            ;reply from the controller
<SO>Y           ;select Y-axis
<ETB>           ;request actual co-ordinates
Y200           ;reply from the controller
<SO>C           ;select all axes
<ETB>           ;request actual co-ordinates
X100Y200Z0      ;reply from the controller
    
```

## CTRL\_O <si> Request Status of Dedicated Inputs

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

### Properties

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

### Description

Immediate instruction **^O = <si> = 0Fh (15d) : REQUEST STATUS OF DEDICATED INPUTS**

The controller continuously updates the 32-bit registers that contain the status of the “DEDICATED INPUTS” of the axes handled by the controller (X, Y, Z, W) and that can be requested at any moment with the code <si> (^O).

The reply from the controller will contain the numerical data that corresponds to the values in the “dedicated inputs” register, preceded by a character that identifies the register in decimal format:

- 'X' : X-axis inputs register
- 'Y' : Y-axis inputs register
- 'Z' : Z-axis inputs register
- 'W' : W-axis inputs register

The data regarding the axes are sent conforming to whatever has been defined using the instruction **^N** for selecting the axes to transmit.

The corresponding bit/input of the individual registers is as follows:

VARIABLE **QIX, QIY, QIZ, QIW** : “X”, “Y”, “Z”, “W” AXIS INPUTS IMAGE

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	<b>07</b>	<b>06</b>	<b>05</b>	<b>04</b>	03	02	01	00
-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----------	-----------	-----------	-----------	----	----	----	----

AXIS DEDICATED INPUTS :	<b>X, Y, Z, W</b>
-------------------------	-------------------

Bit	Description		notes
00	.....	1 = CLOSED  0 = OPEN	
01	.....		
02	.....		
03	.....		
04	Limit switch (-)		
05	Limit switch (+)		
06	Motor drive OK		
07	Machine zero		

E.g.: request inputs image

```
<SI>           ;request status of dedicated inputs
X0 Y32 Z128    ;reply from the controller
```

In the example, the reply sent indicates that:

- 1) the X-axis inputs are all open
- 2) the FORWARD limit switch input is closed on the Y-axis
- 3) the MACHINE ZERO input is closed on the Z-axis

<b>CTRL_P &lt;dle&gt; Annul Instruction in Execution</b>
--

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line; not allowed in the PROGRAMMING mode	

**Description**

Immediate instruction **^P = <DLE> = 10h (16d) : ANNUL INSTRUCTION IN EXECUTION**

With the immediate instruction <DLE> (^P), the instruction being executed can be annulled together with all the instructions waiting to be executed (RESET INSTRUCTIONS FIFO).

Whenever the instruction being executed is a movement, the controller performs a controlled stopping and the annulled movement cannot be restarted and concluded.

Note: The equivalent instruction to the “immediate instruction **^P**”, that works in the “PROGRAMMING” mode is the instruction “**&84**”

E.g.: annul instruction in execution

```
G00 X100      ; instruction to move X-axis to 100
<DLE>        ;annul instruction in execution
<ETB>        ;request actual co-ordinates
X82          ; X-axis co-ordinate reached at the moment in which the movement to 100 was
              annulled
```



If the controller is in ALARM (see instruction <DC2>), the instruction <FF> (^L) must be sent to annul the alarm condition, in addition to the instruction.

**CTRL\_Q <dc1> Xon (enable data transmission)**

**CTRL\_S <dc3> Xoff (suspend data transmission)**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	-	

**Description**

Immediate instruction **^Q = <DC1> = 11h (17d) : XON (ENABLE DATA TRANSMISSION)**

Immediate instruction **^S = <DC3> = 13h (19d) : XOFF (SOSPENDI DATA TRANSMISSION)**

These two codes are used during serial communications to control the flow of transmitted data from the controller towards the Host.

The Host can thus activate or suspend the data transmission from the controller by sending the instructions "XON" and "XOFF".



Even if the transmission is disabled, the controller continues to generate the replies to be sent. Even if enabled, the controller never executes the echo for codes <DC1> and <DC3>.

<b>CTRL_R &lt;dc2&gt; Request Alarm Code</b>
--

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^R = <DC2> = 12h (18d) : REQUEST ALARM CODE**

During the operation, some alarm conditions can be generated which halt the execution of the instructions.

The controller signals the alarm condition by sending the code <BEL> (if enabled: see instruction @85).

With the instruction <DC2> (^R), a message can be requested which indicates the cause of the alarm condition.

For the description of the messages and the causes of an alarm, refer to the description of the **ERROR MESSAGES** in Chapter 2 of this manual.

E.g.: request alarm message

```
<BEL>           ;the controller sends the alarm signal code
<DC2>           ;request to send alarm code
?0-e04         ;alarm code sent by the Cnc
<BEL>           ;alarm signal code
```

**N.B.:** if the alarm has been generated by the PLC program, to the **^R** instruction the controller will reply supplying indication of the nature of the PLC alarm detected.



The alarm condition can be annulled with the instruction **^L = <FF>** if it is dealing with an emergency.

The instruction <DC2> (^R) is accepted even if the controller is not in alarm. In this case no message is sent and the controller merely responds with the PROMPT.

If @82 has been programmed as active (=1), then not only the error code is sent, but also the associated message.

## CTRL\_T <dc4> Single-Block Execution

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line; allowed in the AUTOMATIC mode	

**Description**

Immediate instruction **^T = <DC4> = 14h (20d) : SINGLE-BLOCK EXECUTION**  
 With the code <DC4> (^T) it is possible to execute a program a line at a time (one line for every <DC4> code received).

## CTRL\_U <nak> Repeat Last Message

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^U = <NAK> = 15h (21d) : REPEAT LAST MESSAGE**  
 With the instruction <NAK> the controller is asked to repeat the last string sent. The last string means the group of characters and codes included between the last two PROMPT codes sent by the controller, with the exclusion of any <BEL> code.

E.g.: repeat last string

```
G01 X1000 Y1000
<ETB>           ;request actual co-ordinates
X150 Y150 Z0    ;reply from the controller
<NAK>           ;repeat last message
X150 Y150 Z0    ;reply from the controller
```

## CTRL\_V <syn> Request Dedicated Outputs Status

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

\*\*\*\*

**Description**

Immediate instruction **^V = <SYN> = 16h (22d) : REQUEST DEDICATED OUTPUTS STATUS**  
 the controller continuously updates the 32-bit register that contains the “dedicated output” statuses of the axes handled by the board (X, Y, Z, W) and that can be requested at any moment by the code <SYN> (^V).

The reply from the board will contain the numerical data in decimal format that corresponds to the values in the “dedicated outputs” register, preceded by a character that identifies the register:

- 'X' : X-axis inputs register
- 'Y' : Y-axis inputs register
- 'Z' : Z-axis inputs register
- 'W' : W-axis inputs register

The data relative to the axes are sent in conformity with those defined by the instruction **^N** for selecting the axes to transmit.

The corresponding bit/output for the individual registers is as follows:

VARIABLES **QOX, QOY, QOZ, QOW** : AXIS OUTPUTS IMAGE “X”, “Y”, “Z”, “W”

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	<b>07</b>	<b>06</b>	<b>05</b>	<b>04</b>	<b>03</b>	<b>02</b>	<b>01</b>	<b>00</b>
-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

AXIS dedicated outputs : <b>X, Y, Z, W</b>
--

Bit	Description	Notes
00	Motor drive direction	1 = ACTIVE  0 = DISACTIVE
01	....	
02	....	
03	....	
04	....	
05	“Encode zero” polarity	
06	“Machine zero” polarity	
07	Set point enabled	

E.g.: request status of outputs

```
<SYN>           ;request status of outputs
X0 Y33 Z133     ;reply from the controller: status of outputs
```

<b>CTRL_W &lt;etb&gt; Request Actual Co-ordinates</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^W = <ETB> = 17h (23d) : REQUEST ACTUAL CO-ORDINATES**

With the instruction <ETB> (^W) it is possible to ask the controller to send the values of the actual co-ordinates of the axes, even if the axes are in movement.

The co-ordinates sent by the board will be in conformity with the settings made with the customisation instructions @71 and G70-G71 for defining the number of decimal digits and the units of measurement.

The corresponding variables that contain the transmitted data are: **QRX, QRY, QRZ, QRW**

E.g.: request actual co-ordinates

```
<ETB>                ;request actual co-ordinates
X-38.56 Y100.00 Z1.50 ;reply from the controller: actual co-ordinates
```



The format of the reply from the controller depends on the selections made with the code “^N <SO>”. If the CNC is selected, the co-ordinates are always sent in the order of the axes X, Y , Z and W (if handled).

The co-ordinate sent is always an ABSOLUTE co-ordinate, which means referred to the absolute origin active at that moment.



**CTRL\_X <can> Cancel String Sent**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	-	

**Description**

Immediate instruction **^X = <CAN> = 18h (24d) : CANCEL STRING SENT**

The code <CAN> (^X) can be used to cancel all the previous characters sent as a line of instructions.

Naturally the line of instructions must not yet have been closed with the code <CR>.

The code <CAN> sent as the first piece of data in a line of instructions has no effect.

E.g.: cancel a line of characters

Correction of a line of instructions:

@01 X100<CAN>G00X100

The instruction "@01X100 is cancelled and the only instruction accepted by the CNC is: G00 X100

<b>CTRL_Y &lt;em&gt; Reset Cnc (software)</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction **^Y = <EM> = 19h (25d) :     **RESET CNC (SOFTWARE)****

The instruction <EM> (^Y) is particular as it requires the sending of a further character before it can be executed.

When this instruction is received, the controller sends the character '!' and places itself in hold expecting a subsequent character in reply as **confirmation of the instruction**.

If the character '**S**' is received, then the instruction will be executed.

The sending of any other character or code will annul the instruction <EM>.

The execution of the instruction <EM> is equivalent to powering up the controller again. Therefore, after the receipt of the PROMPT, the procedure for re-establishing communications must be repeated.

E.g.: reset CNC

```

<EM>           ;instruction of Reset Cnc
!              ;request confirmation from the controller
S              ; confirmation of the instruction to Reset Cnc
    
```



The instruction <EM> is executed, if confirmed, whatever state the controller is in: emergency, alarm, axes stationary or in movement.

<b>CTRL_Z &lt;sub&gt; Synchronisation Code</b>
--

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line; not allowed in the PROGRAMMING mode	

**Description**

Immediate instruction **^Z = <SUB> = 1Ah (26d) : SYNCHRONISATION CODE**

The instruction <SUB> (^Z) can be used to synchronise the user program with the execution of the instructions. It is always used coupled with the function G65 (G66) of "await synchronisation code".

In practice, it represents an "OK" signal that is sent to the controller to give consensus for it to execute the instruction that follows the code G65 (G66).

E.g.: synchronisation code

```
G00 X100
G65 O1      (the controller will not proceed with execution until ● )
G00 X100
<SUB> ←-----
```



The code <sub> must be sent only when the function G65 (G66) is in execution.

**CTRL \ <fs> Request Theoretical Co-ordinates**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line	

**Description**

Immediate instruction  $\wedge$  = <FS> = 1Ch (28d) : **REQUEST THEORETICAL CO-ORDINATES**  
 With the instruction <FS> ( $\wedge$ ) it is possible to ask the controller to send the THEORETICAL co-ordinates of the axes, even when they are in movement.  
 The co-ordinates sent by the board will conform to the format defined with the customisation instruction @71 to define the number of decimal digits, and the instructions G70 and G71.  
 The corresponding variables that contain the transmitted data are: **QTX, QTY, QTZ, QTW**

E.g.: request theoretical co-ordinates

```
<FS>           ;request to send theoretical co-ordinates
X-38.56 Y100.00 Z1.50 ;reply from the controller: theoretical co-ordinates
```



The format of the reply sent by the controller depends on the selection made with the code “ ^N <SO>”. If the CNC is selected, the co-ordinates of the axes are always sent in the order X, Y,Z and W (if handled).  
 The co-ordinate sent is always an ABSOLUTE co-ordinate, which means referred to the absolute origin that is active at that moment.

**CTRL\_] <gs> Stop Cnc at End of Block**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	First character of the line; not allowed in the PROGRAMMING mode	

**Description**

Immediate instruction **^]** = **<GS>** = **1Dh (29d)** : **STOP CNC AT END OF BLOCK**

With the code **<GS>** (immediate instruction **^]**) it is possible to interrupt the execution at the end of the instruction being executed.

The STOP condition is indicated by the relative bit in the status register.

The execution of the instructions can be subsequently continued by sending the code **<ETX>** (**^C=START**).

E.g.: stop CNC at end of block

```
G00 X0 Y0
G00 X100 ←-----]
G00 Y100
<GS>      (the controller terminates the line in execution and stops )
<ETB>
X100 Y0
```





# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 4

-

“@” and “%”

## Characterisation Instructions





## CONTENTS

Chapter 4 - “@” and “%” Characterisation Instructions.....	4-1
Introduction.....	4-4
@01 ÷ %01 Define “Programmable” Digital Inputs .....	4-6
@03..@06 ÷ %03..%06 Define “Dedicated” Digital Inputs .....	4-9
@15 / %15 Define " Encoder Zero" input.....	4-11
@16 / %16 Define Direction of Co-ordinates Count.....	4-12
@20 / %20 Define Cam Table Parameters.....	4-13
@21 / %21 Define Cam Type and Number of Repetitions.....	4-14
@22 / %22 Define “Programmable” Digital Outputs .....	4-15
@30 / %30 Define I/O Devices on CAN BUS .....	4-17
@31 / %31 Define "Dir" output (Motor Drive Direction).....	4-18
@32 / %32 Define "Enb" output (Enable Motor Drive).....	4-19
@33 / %33 Define Analogue Output Type.....	4-21
@34 / %34 Define Polarity Analogue output.....	4-22
@35 / %35 Defining Output Polarity in Frequency.....	4-23
@40..@49 Define Servocontrol Parameters.....	4-24
@40 / %40 Define Limit Levels .....	4-25
@41 / %41 Define Dead Band .....	4-26
@42 / %42 Define 2nd Servo Proportional Gain.....	4-27
@43 / %43 Define Servo Integral Gain.....	4-28
@44 / %44 Define Servo Derivative Gain.....	4-29
@45 / %45 Define Derivative Sampling Time of the Servo.....	4-30
@46 / %46 Define Feedforward Percentage .....	4-31
@47 / %47 Define Servo Integral Limit.....	4-32
@48/%48 Define Level between the 1st and 2nd Proportional Gain.....	4-33
@49 / %49 Define 1° Servo proportional gain .....	4-34
@50 / %50 Define Space-Pulse Ratio of the Transducer .....	4-35
@51 / %51 Define Space-Step Ratio of the Motor.....	4-36
@52 / %52 Define Analogue voltage ratio .....	4-37
F@53 / %53 Define Maximum Frequency .....	4-38
@54 / %54 Programming Velocity Limit .....	4-39
@56 / %56 Define Velocity / Frequency of Start-Stop .....	4-40
@57 / %57 Define Acceleration/Deceleration Ramp .....	4-41
@59 / %59 Define Level per Axis in Position.....	4-42
@60-@61 / %60-%61 Define Machine Limits.....	4-43
@70 / %70 Define Panel key active states .....	4-45
@71 / %71 Define Data Format Sent by the Controller .....	4-47
@72 / %72 Define Type of Axis: Linear / Angular.....	4-48
@80 / %80 Enable the Percentage Variation of the Axes' Velocity.....	4-49
%81 Read Variables.....	4-50
@82 / %82 Reserved Instruction.....	4-51
@83 / %83 Enable Line in Execution .....	4-52
@84 / %84 Enable Velocity Limit.....	4-53
@85 / %85 Enable <BEL> Code Transmission .....	4-54
@86 / %86 Modify the Communication Protocol (PLC) .....	4-55
@90 / %90 Define Error Level of Co-ordinates of the Centre .....	4-56
@91 / %91 Define “Objective Reached” Level.....	4-57
@94 , %94 Define Tangential Tool "level".....	4-58
@95 , %95 Define " no. of cutting edges " of the tangential tool .....	4-59
@96 / %96 Define tangential tool "raise/lower times" .....	4-60
@97 / %97 Enable Control Panel on Power-up.....	4-61
@98 Recover Parameters from Non-volatile Memory.....	4-62
%98 Read Configuration of the Control System .....	4-63
@99 Save Parameters in the Non-volatile Memory .....	4-65
%99 Read Software Version .....	4-66

## INTRODUCTION

### @ FUNCTIONS

The '@' functions are codes for programming a series of characteristic parameters of the system, enabling the control functions to be adapted to the application.

For this reason they are known as CHARACTERISATION CODES.

All the data programmed using '@' functions can be saved in a non-volatile memory in the controller. In this way, the controller will be ready to work correctly, each time it is powered up, without further programming of these parameters.

### % FUNCTIONS

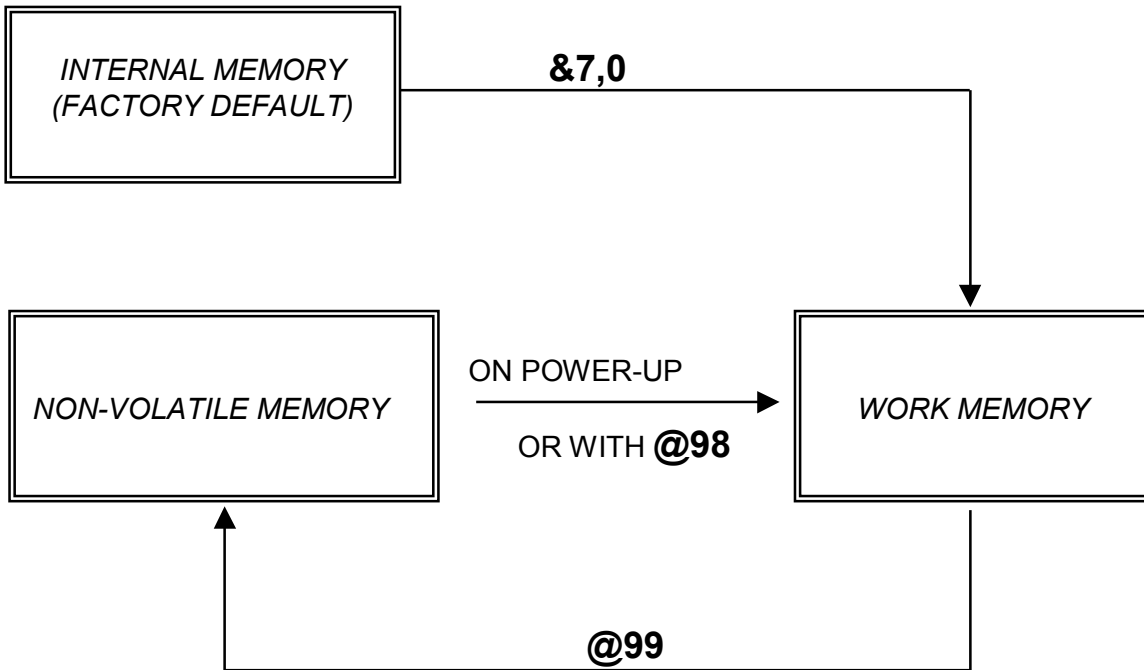
The '%' functions are dedicated to the reading of the system parameters that have been programmed with the '@' codes.

To every '@' function, used for defining parameters, there is a corresponding '%' function to enable the system configuration to be read.

Apart from the codes for reading these parameters, there is the special %99 function that reads the version of software installed in the control system.

The % instructions can be sent without specifying the axis: in which case all the axes will be displayed.

## ORGANISATION OF "CNC-MEMORY" CHARACTERISATION PARAMETERS



The system uses three different types of memory during its operation:

- INTERNAL MEMORY (factory default)
- WORK MEMORY
- NON-VOLATILE MEMORY

The internal memory stores the factory default parameters, that may subsequently be re-defined using the @ functions. The first time the unit is powered up, these are recalled automatically and loaded into the work memory (that used by the CNC) and can be saved in the non-volatile memory with any eventual modifications, using the instruction @99. Subsequently, the default parameters can be recalled when required using the function &7,0.

On every subsequent power-up, the controller will load the parameters stored in the non-volatile memory into the work memory. They can be recalled at any moment with the function @98.

In any case, any modification that is to be made to the non-volatile memory must first be performed in the work memory and then saved with the function @99.

## @01 ÷ %01 Define “PROGRAMMABLE” Digital Inputs

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@01 In,(s) ,(f) [In,(s),(f)] [...] In; digital input no.(1...16) (s); define input state (f); define associated function	(1)
% syntax	%01	(1)
CNC reply	In,(s),(f)	

Notes: (1) The programming only concerns the digital inputs defined in the command

**Description**

The possibility of programming one or more of the digital inputs present on the controller board such that it is associated with a function whose execution is then automatically handled by the controller itself. The instruction for programming the digital inputs is “@01” which must be followed by the address letter “I” and the number “n” of the input that it is wished to program, the separator “,” followed by the parameter “s” that defines the active state of the input and the parameter “f” that defines the function associated with it.

In: NUMBER OF DIGITAL INPUT OF THE CONTROLLER: 1...16

(S) : 0 = ENABLED; ACTIVE LOW (OPEN CONTACT / OPENING CONTACT)  
 1 = ENABLED; ACTIVE HIGH (CLOSED CONTACT / OPENING CONTACT)  
 2 = DISABLED (NOT PROGRAMMED)

(f) : 00 = EMERGENCY ; active on front  
 01 = START/STOP ; active during status  
 02 = STOP AT END OF BLOCK ; active on front  
 03 = STOP IMMEDIATELY ; active on front  
 04 = START ; active on front  
 05 = SET/RESET EMERGENCY ; active during status  
 06 = ANNUL INSTRUCTION ; active on front  
 07 = AWAIT TANG. TOOL DOWN ; active during status  
 08 = AWAIT TANG. TOOL UP ; active during status  
 09 = EMERGENCY RESET ; ACTIVE ON FRONT  
 10 = IMMEDIATE STOP ; ACTIVE DURING STATUS

ACTIVATION “on front”;

Means that the associated function is executed when the input is switched from one state to the other:

- switch 1→0 (closed contact → open contact) if input is active low
- switch 0→1 (open contact → closed contact) if input is active high

ACTIVATION “during status”;

Means that the associated function is executed when the input is in that programmed state:

- status 0=open contact
- status 1=closed contact

## Description of the FUNCTIONS

The function defines the type of operation performed automatically by the controller at the moment in which the input finds itself in the state programmed as active. The functions that may be associated with inputs are:

### **EMERGENCY** (active on front)

The activation of the input puts the controller in emergency. Disactivating the input, the controller remains in emergency.

### **START/STOP** (active during status)

The activation of the input will cause the controller to stop immediately (stop the axes with controlled braking) and maintain a constant monitoring of the stop condition as its disactivation will be its signal to start again.

### **STOP at END OF BLOCK** (active on front)

The activation of the input causes the controller to stop when it has completed its current instruction. Disactivating the input, the controller remains in the stop condition.

### **IMMEDIATE STOP** (active on front)

The activation of the input causes the controller to stop immediately (stop axes with controlled braking). Disactivating the input, the controller remains in the stop condition.

### **START** (active on front)

The activation of the input re-starts the controller (if in the stop condition). Its disactivation has no effect.

### **SET / RESET EMERGENCY** (active during status)

The activation of the input puts the controller in emergency while its disactivation resets the state of emergency.

The effect of the input occurs “during the status”, that is while the input is in its active state, the emergency condition can only occur by disactivating it. While it is active, the immediate instruction Ctr-L will have no effect. If the input is not active, it is still possible to put the controller in emergency with the immediate instruction Ctrl-K (set emergency).

### **ANNUL INSTRUCTION** (active on front)

The activation of the input causes immediate annulment of the instruction in execution and all the instructions in the queue to be executed (reset the instructions FIFO). Any movement taking place will come to a controlled stop.

The activation of the input is equivalent to the execution of the immediate instruction Ctrl-P and it can therefore be used for the annulment of the execution of a program in the AUTOMATIC and JOG modes or in an alarm condition. The disactivation of an input programmed in this way will have no effect.

### **WAIT FOR TANGENTIAL TOOL “DOWN”** (active during status)

In the models used for the handling of the TANGENTIAL TOOL, this enables the TOOL-DOWN input to be controlled so that, during an operation in which the tool is lowered automatically by the CNC when it is active, and the function G93 (tangential tool guide ON), the interpolated movement of the two axes on the plane do not re-start until the tool position has been confirmed as down (input active).

### **WAIT FOR TANGENTIAL TOOL “UP”** (active during status)

In the models used for the handling of the TANGENTIAL TOOL, this enables the TOOL-UP input to be controlled so that, during an operation in which the tool is raised automatically by the CNC when it is active, and the function G93 (tangential tool guide ON), the rotation of the axis of the tangential tool does not begin until the tool position has been confirmed as up (input active).

**EMERGENCY RESET** (active on front)

The valid switching of the signal will provoke the emergency reset, on the condition that a cause for the condition no longer exists.

**IMMEDIATE STOP** (active during status)

The activation of the input will provoke the immediate stop of the controller (stop the axes with controlled braking). While the input is active, the start instruction from whatever source will be ignored. Once the input has been deactivated, the controller will remain in the stop condition until a successive start.

**N. B. :**

- If there is no active PLC program on the controller, the “programmed” inputs, these are those with associated functions handled automatically by the controller, can still be read (and therefore used) by the CNC program, using the **G69** instruction (e.g. **G69 P1** reads the state of input 1). If, on the other hand, the controller has a resident PLC program, the interchange bits between the PLC and CNC are read with the instruction **G69 P(n)**. Obviously the PLC program can copy the states of the inputs that must be visible to the CNC in the corresponding interchange bits, letting the program be handled once more by the CNC program.
- Remember that of the all the possible inputs, there are “dedicated” inputs that have functions pre-assigned by the controller (limit switch and/or machine zero). Therefore if these are programmed again by the user, they will have a double function, which could create a conflict condition. The programmer must be careful to avoid any possible conditions of conflict when programming these inputs.

In any case, the priority of execution, should there be multi-programming of an input, is:

- 1st - function pre-assigned by the controller
- 2nd - function programmed to a digital input
- 3rd - function activated by a key on the control panel

e.g.: define programmable digital input no.1

```
@01 I1,0,0           ;define digital input no.1 as a “programmed” input, active low,
                    ;with the associated function = 0 (EMERGENCY)
```

e.g.: read programmed digital input no.2

```
%01 I               ;request to read
I2,1,1             ;reply from controller (input 2 enabled, active high, with strt/stop function))
```

**N:B.:** in reply to the **%01** instructions, the controller omits information about the disabled inputs.

**@03..@06 ÷ %03..%06 Define “DEDICATED” Digital Inputs**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@03 [X(i)] [Y(i)] [Z(i)] [W(i)] @04 [X(i)] [Y(i)] [Z(i)] [W(i)] @05 [X(i)] [Y(i)] [Z(i)] [W(i)] @06 [X(i)] [Y(i)] [Z(i)] [W(i)] X(i); Y(i); Z(i); W(i); axis input; range 0..2 The value (i) can have the following values: 0 = enabled, active low 1 = enabled, active high 2 = disabled (not allowed for the instruction @06)	(1)
% syntax	%03 [X] [Y] [Z] [W] %04 [X] [Y] [Z] [W] %05 [X] [Y] [Z] [W] %06 [X] [Y] [Z] [W] X,Y,Z,W: input condition	(1)
CNC reply	[X(i)] [Y(i)] [Z(i)] [W(i)]	

Notes: (1) La programming only concerns axes inputs defined in the instruction  
 The addresses Y, Z and W are allowed only if they are available on the model of controller selected

**Description**

The controller can handle 4 inputs with predefined (dedicated) functions for each axis:

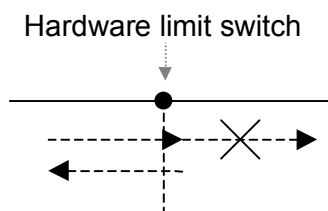
- LIMIT SWITCH FORWARD (@03)
- LIMIT SWITCH BACKWARD (@04)
- MOTOR DRIVE O.K. (FAULT) (@05)
- MACHINE ZERO (@06)

With the functions @03..@06 it is possible to program the active status of each input and enable or disable the input itself.

**Important**

If the dedicated inputs with the functions of limit switch and motor drive o.k. become active, the controller goes into emergency (if enabled).

If the hardware limit switches are employed (@03 e @04), the controller goes into emergency and the motor drive is disabled. Once the controller reset is performed, the latter will move only in the direction away from the limit switch (see figure). Any further attempt to move beyond the limit switch will provoke a new emergency condition.



Programming the input with the function of machine zero depends on the status that the input must assume during the Set Point procedure (see function G51).

If this input is disabled, the Set Point procedure will be performed considering only the zero input of the transducer (ruler or encoder).

e.g.: define dedicated inputs

```
@03 X1 Y1      ;programming LIMIT SWITCH FORWARD X- and Y-axes: enabled, active high
@04 Y0 X0 Z0   ;programming LIMIT SWITCH BACKWARD X-,Y- and Z-axes: enabled, active low
```

e.g.: read

```
%06 Z          ;request to read programmed input of machine zero of the Z-axis
Z1             ;reply from controller
```

```
%03 YX        ;request to read LIMIT SWITCH FORWARD X- and Y-axes
X1 Y0         ;reply from controller
```



**@15 / %15 Define " ENCODER ZERO" input**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

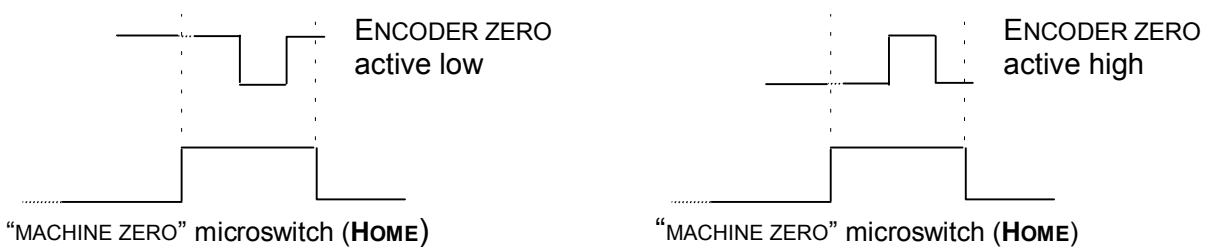
Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@15 [X(z)] [Y(z)] [Z(z)] [W(z)] X,Y,Z,W: encoder zero; range 0..1 The value (z) can have the values: 0 = active low; 1 = active high	(1)
% syntax	%15 [X] [Y] [Z] [W] X,Y,Z,W: transducer zero; range 0..1	(1)
CNC reply	[X(z)] [Y(z)] [Z(z)] [W(z)]	

Notes: (1) The programming concerns only the inputs of the axes defined in the instruction.  
The addresses Y, Z and W are allowed only if available on the model of controller chosen.

**Description**

The function @15 is used to define the active state of the "ENCODER ZERO" coming from the external incremental encoder that is used together with the "HOME" input, for each axis handled by the controller, during the Set Point procedure (seeking the machine zero).

(z) : active state "ENCODER ZERO"  
0 = active low;  
1 = active high;



e.g.: programming the active state of "ENCODER ZERO" for X,- Y- and Z-axes

@15 Y1 X1 Z0 ;the active state of the "ENCODER ZERO" signal for the X- and Y-axes is set as active high and that of the Z-axis as active low.

e.g.: reading active state of the "ENCODER ZERO" input

%15 XYZ ;request to read the active state of the "ENCODER ZERO" input  
X0 Y1 Z0 ;reply from controller

## @16 / %16 Define Direction of Co-ordinates Count

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@16 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: count direction; range 0..1 The value (n) can have the values: 0 = count active "normal" 1 = count active "reverse"	<b>(1)</b>
% syntax	%16 [X] [Y] [Z] [W] X,Y,Z,W: count direction; range 0..1	<b>(1)</b>
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

This instruction is used to define the count direction of the co-ordinates by programming the logic values 0 and 1 for each axis handled.

If, after connecting the phase signals of the transducer to the CNC, the count direction is not correct (for example if moving the axis forward (+), the co-ordinate value decrements), it is possible to use the function @16 to invert the count direction without physically inverting the wires of the phase signals from the encoder to the controller.

This instruction has no meaning for stepper motors without feedback.

(n) : active state of "ENCODER COUNTER"  
0 = active "normal";  
1 = active "reverse";

e.g.: define count direction of the X, Y and Z axes

@16 X0 Y1 Z0 ;the count direction of the X- and Z-axes are defined as "normal", and the Y-axis is defined as "reverse"

e.g.: read direction of count

%16 XYZ ;request to read "logic" direction of the co-ordinate count  
X0 Y1 Z0 ;reply from controller

## @20 / %20 Define Cam Table Parameters

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@20E<n.tab>,<n.points>,<spline points >,<M><S0>..(Sn) n.tab = number of table to set up n.points = number of points that make up the positions table spline points = number of points that make up the spline M = identifier of master axis S0 = identifier of slave axis 0 Sn = further identifiers of slave axes	<b>(1)</b>
% syntax	%20	
CNC reply	E0<n.points>,<spline points>,<M><S0>..(Sn) E1<n.points>,<spline points>,<M>,<S0>..(Sn)	<b>(2)</b>

- Notes:
- (1) If no parameter is given after 'n. tab', the table is disallocated
  - (2) The parameters of both tables are always displayed

**Description**

The instruction initialises all the parameters regarding the cams and allocates the memory required for defining the points table.

e.g.: define cam table parameters

@20E0,50,0,ZXY ;define the table '0' with '50' points, '0' spline points, with the Z-axis defined as master and the X- and Y-axes as slaves

e.g.: read cam table parameters

%20 ;request to read the parameters of the table  
 E0,50,0,ZXY E1,0,0,XXX ;reply from controller (E1 has not been defined)

**Note**

The instruction is available only if the controller is installed with this option

## @21 / %21 Define Cam Type and Number of Repetitions

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

### Properties

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@21E<n.tab>,<T>,(N) n.tab = table number T = cam type:           0 = position 1 = absolute velocity (not active) 2 = percentage velocity (not active) N = number of repetitions of the table	<b>(1)</b>
% syntax	%21	
CNC reply	E0,<T>,(N) E1,<T>,(N)	

Notes: (1) If N is set to '0' the controller executes an infinite number of repetitions

### Description

The instruction defines the cam type that is to be used and the number of repetitions of the table.

e.g.: define cam type and number of repetitions

```
@21E0,1,3 ;for table '0' defines cam type as 'absolute velocity' with 3 repetitions
           of the table
```

e.g.: read cam type and number of repetitions

```
%21 ;request to read cam type and number of repetitions
E0,1,3 E1,0,0 ;reply from controller
```

### Note

The instruction is valid only if the controller is installed with the appropriate option

## @22 / %22 Define “Programmable” Digital Outputs

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@22 On,(s) ,(f) On; n. digital output (1...8) (s); define output status (f); define associated function	<b>(1)</b>
% syntax	%22 On On; n. digital output (1...8)	<b>(1)</b>
CNC reply	On,(s),(f)	

Notes: (1) The programming concerns only the digital outputs defined in the instruction

**Description**

There is the possibility of programming one or more of the digital outputs available on the controller such that they are associated with a function whose execution is then automatic and handled by the controller itself.

The instruction for programming the digital outputs is “@22” and it must be followed by the address letter “O” and the number “n” of the output that is required to be programmed, separated by a “,” the parameter “s” which defines the active state of the output must follow and finally the parameter “f” that defines the function to be associated with it.

On: NUMBER OF DIGITAL OUTPUT OF THE CONTROLLER: 1...8

(S) : 0 = ENABLED; ACTIVE LOW (OPEN CONTACT)  
 1 = ENABLED; ACTIVE HIGH (CLOSED CONTACT)  
 2 = DISABLED (NOT PROGRAMMED)

(f) : 0 = EMERGENCY  
 1 = STOP WITH AUTORESET  
 2 = EMERGENCY STATUS  
 3 = STOP WITHOUT AUTORESET  
 4 = RAISE/LOWER TANG. TOOL  
 5 = .....  
 6 = .....

**Description of FUNCTIONS**

The function defines the type of operation performed automatically by the controller at the moment that the status becomes active.

**EMERGENCY**

When the controller goes into Emergency, this output is activated.

**STOP WITH AUTORESET**

When the controller is in this Stop condition, the output is activated. When the Stop condition ends, the output disactivates.

**STATUS OF EMERGENCY**

The indicated status is signalled at the specified output when the controller is in Emergency. If the controller is not in Emergency, the status can be assigned by the operator, by the CNC program or by the PLC program.

**STOP WITHOUT AUTORESET**

When the controller is in this Stop condition, the output goes active and remains active even after the Stop condition has ended. The operator, the CNC program or the PLC program can assign a different state.

**RAISR/LOWER TANGENTIAL TOOL**

The output is handled automatically to pilot the approach of the tool in conformity with what has been defined by the **@94**, **@95** and **@96** instructions.

**N. B.:**

- Apart from the outputs defined as EMERGENCY and STOP WITH AUTORESET, the “programmed” outputs, these are those with associated functions handled automatically by the controller, can be written to (and therefore used) by the CNC, using the **G67(G68)** instruction.

e.g.: define programmable digital output n.1

```
@22 O1,0,0 ;define digital output n.1 as “programmed” output, active low,
;with associated function = 0 (EMERGENCY)
```

e.g.: read programmed digital output n.2

```
%22 ;request to read
O2,1,1 ;reply from controller (output 2 enabled high with the associated function of Stop
without Autoreset)
```

## @30 / %30 Define I/O Devices on CAN BUS

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@30 In,(t) ,(i) In,(t),(i) In,(t),(i), ..... In; no. of I/O device (1...16) (t); device type (i); device address	<b>(1)</b>
% syntax	%30 In; device no. (1...16)	<b>(1)</b>
CNC reply	In,(t),(i) In,(t),(i) In,(t),(i),.....	

Notes: (1) It is necessary to program all devices in the same instruction.  
To eliminate all the devices, execute @30 without parametres

**Description**

It is possible to handle one or more I/O device non CAN BUS connected to the controller.  
The instruction for programming the devices is “@30” and it must be followed by the letter “I” and the logic number of the device, by the type “t” of the device that is to be added followed by the parameter “i” that identifies the address of the board.

In: LOGIC NUMBER OF THE I/O DEVICE : 1...16

- (t): 0 = CBM\_00 BOARD (8 IN, 8 OUT)
- 1 = CBM\_01 BOARD (16 IN)
- 2 = CMB\_02 BOARD (16 OUT)
- 8 = CMB\_03 BOARD (24 IN, 16 OUT)

(i): address of board is determined by the configuration of jumpers on the board.

For more information refer to the documentation associated with the board.

**Note**

The instruction is valid only if the controller is installed with the appropriate option

**@31 / %31 Define "DIR" output (Motor Drive Direction)**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@31 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: motor drive direction; range 0..2 The value (n) can have the values: 0 = output "DIR" active at logic level 0 1 = output "DIR" active at logic level 1 2 = output "DIR" disabled	(1)
% syntax	%31 [X] [Y] [Z] [W] X,Y,Z,W: motor drive direction; range 0..2	(1)
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if they are available on the model of the controller chosen.

**Description**

The "DIR" output for the motor drive direction, (forward/backward) of each axis handled by the controller, can be programmed on the CNC at logic level 1 or 0.

Thanks to this function, it is possible to define (for motor drives that use this signal) the direction of the movement of the axis as a function of the movement instruction forward (+) or backward (-) of the axis itself.

- (n) : active state "DIR : MOTOR DRIVE DIRECTION"
- 0 = "DIR" output active at logic level 0;
- 1 = "DIR" output active at logic level 1;
- 2 = "DIR" output disabled

e.g.: define motor drive of the X- and W-axes

@31 X1 W1 ;the X- and W-axes are programmed with direction outputs active at level '1'

e.g.: read motor drive direction

%31 XW ;request to read motor drive direction type  
X1 W1 ;reply from controller



**@32 / %32 Define "ENB" output (Enable Motor Drive)**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@32 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: enable motor drive; range 0..2 The value (n) can have the values: 0 = "ENB" output active at logic level0 1 = "ENB" output active at logic level1 2 = "ENB" output disabled	(1)
% syntax	%32 [X] [Y] [Z] [W] X,Y,Z,W: enable motor drive; range 0..2	(1)
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

Define enable motor drive (logic of the Enable or Disable signals).  
For the controller, the "Enable" output is active when the motor drive can move the axis and is inactive when the motor drive is disabled, for example in the emergency condition.  
The selection made with this function (@32) makes it possible to adapt the behaviour of this signal from the CNC (active/inactive state) according to the motor drive fitted.

The string "@32X2" implies that the relative signal is disabled and therefore left open.

For the Goya controller only, the following table applies.

Parameter	Enable	Output state
@32X0	Active	Closed to GND
	Inactive	Open
@32X1	Active	Open
	Inactive	Closed to GND

In other cases, see the following table.

Parameter	Enable	Output state
@32X0	Active	Open
	Inactive	Closed to GND
@32X1	Active	Closed to GND
	Inactive	Open

e.g.: define enable motor drive X- and Y-axes

```
@32 X1 Y1           ;the X- and Y-axes are programmed with enable outputs active a  
                    ;level '1'
```

e.g.: read enable motor drive

```
%32 XY             ;request to read enable motor drive type  
X1 Y1              ;reply from controller
```

## @33 / %33 Define Analogue Output Type

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@33 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: type of analogue output ; range 0 or 1 The value (n) can have the values: 0 = bipolar analogue output (-10/+10) 1 = unipolar analogue output (0/+10)	<b>(1)</b>
% syntax	%33 [X] [Y] [Z] [W] X,Y,Z,W: : type of analogue output; range 0 o 1	<b>(1)</b>
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
 The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

The function @33 is used to select the type of analogue output handled by the controller for each axis. It can be bipolar (-10/+10 Volt) or unipolar (0/+10 Volt).  
 The code is meaningful for axes driven by d.c. motors.

e.g.: define type of analogue output

@33 X0 Y0 W0 ;The X-, Y- and W-axes are programmed with bipolar analogue outputs

e.g.: read type of analogue output

%33 X Y W ;request to read type of analogue output  
 X0 Y0 W0 ;reply from controller

**N.B.:** in the case of the Goya controller that pilots a stepper motor drive, the relative analogue output to that axis can be used only in bipola mode, otherwise there is a conflict with the DIR signal handling.

## @34 / %34 Define Polarity Analogue output

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@34 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: polarity of analogue output ; range 0..1 The value (n) can have the values: 0 = positive voltage: axis backward (-) 1 = positive voltage: axis forward (+)	<b>(1)</b>
% syntax	%34 [X] [Y] [Z] [W] X,Y,Z,W: : polarity of analogue output ; range 0..1	<b>(1)</b>
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
 The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

For bipolar analogue outputs, it is possible to define, using the function @34, the direction of the displacement of the axis (forward or backward) that is obtained for a positive voltage. Obviously the code is meaningful for axes driven by d.c. motors.

e.g.: define analogue output polarity

@34 X1 Y1 ;The X- and Y-axes move in the forward direction for a positive voltage

e.g.: read analogue output polarity

%34 XY ;request to read polarity of analogue output  
 X1 Y1 ;reply from controller

## @35 / %35 Defining Output Polarity in Frequency

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the program block	
Execution	Queued	
Condizions	-	
Compatible Functions	-	
Syntax @	@35 [X(n)] [Y(n)] [Z(n)] [W(n)] X,Y,Z,W: output polarity in frequency; range 0..1 The value (n) assumes the meaning: 0 = pulse towards the positive voltage (+) 1 = pulse towards 0 V (-)	<b>(1)</b>
Syntax %	%35 [X] [Y] [Z] [W] X,Y,Z,W: : output polarity in frequency; range 0..1	<b>(1)</b>
CNC Reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming only concerns the axes defined in the instruction. The addresses Y, Z and W are allowed only if available on the model of controller chosen.

**Description**

With a frequency output, that is with a stepper motor, using function @35, it is possible to define the polarity of the pulse that is sent to the drive unit. Most stepper motor drive units require an input signal which is normally held at 0V and which is pulsed positive to perform a step of the motor. Nevertheless, there are some drive units with the opposite operation: the input being normally held at the positive voltage and step the motor when they receive a pulse towards 0V. Obviously this entry is only meaningful for axes driven by stepper motors.

Es.: defining output polarity in frequency

@35 X1 Y0 ;the X-axis will be controlled by pulses towards 0V and the Y-axis by pulses towards the positive voltage

Es.: read output polarity in frequency

%35 XY ;request to read the output polarity in frequency  
X1 Y0 ;reply of the controller

## @40..@49 DEFINE SERVOCONTROL PARAMETERS

The purpose of this section is to explain how the servocontrol mechanism of the axes driven by d.c. motors functions.

For these types of axes, a **POSITION CONTROL** is achieved using the signals from the transducer (encoder or optical ruler) where a continual comparison is made between the **theoretical** and the **actual** position of the axis.

If the positions do not coincide, that is that there is a **position error**, the controller acts on the motor speed, regulating the output voltage signal of the controller with the aim of eliminating the error in the position.

The controller instantly calculates the **theoretical values** of the position and the velocity expressed in steps and corrects the latter as a function of the **position error** (difference between the theoretical and actual positions) and the **gain** or a numerical value that determines the amount of correction to apply.

The need to configure the servocontrol mechanism arises, obviously, from the differences in the possible applications.

The parameters that may be defined with the functions @40...@49 are used to **tune** the servocontrol mechanism for its application.

Each function is described individually in the following section as far as regards syntax, addresses, units of measurement and limit values.

In all S&h controllers, the output signal is the algebraic sum of four components: “feedforward”, proportional correction, and integral correction, calculated according to the following guidelines:

- Open loop gain (feedforward): the component proportional to the velocity required that is present even in the case of perfect coincidence (actual and historical) between the theoretical and real position ( $\varepsilon = 0 \cap d\varepsilon / dt = 0 \cap \int \varepsilon dt = 0$ ).
- Dead band (nr): the maximum error between the real and theoretical position that is ignored by the control loop.
- First proportional action (Kp1): the component proportional to the instantaneous error that manifests itself up to a maximum error defined by the user, the so-called “knee point” (Gp).
- Second proportional action (Kp2): the component proportional to the instantaneous error that manifests itself beyond the “knee point”,
- Derivative action (Kd): the component proportional to the difference between the current error and the previous error measured in a time defined by the user, the so-called “sampling time” (st).
- Integral action (Ki): the component proportional to the algebraic sum of all the errors verified during the operation. This component has a limit of an absolute value, the so-called “integral limit” (ib) defined by the user.

One last parameter at the discretion of the user is the maximum error inside which the controller tries to recover, the so-called “servo error limit” (se). For instantaneous errors greater than this setting, the controller renounces its task of following and goes into emergency.

The open loop gain component is defined by the following formula

$$FFO = \frac{Kff}{100} \frac{V}{Vmax} \text{ OUTmax,}$$

Where OUTmax is the maximum output voltage that the controller can generate, V is the velocity requires at this instant, Vmax is the maximum velocity at which the axis can move and Kff is the proportion of feedforward.

### Note

Refer to the chapter F of the Manual’s Appendix, for another information

## @40 / %40 Define Limit Levels

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@40 [X(level)] [Y(level)] [Z(level)] [W(level)] X(level),Y(level),Z(level),W(level): level of error	(1) (2)
% syntax	%40 [X] [Y] [Z] [W] X,Y,Z,W: : level of error	(1)
CNC reply	[X(level)] [Y(level)] [Z(level)] [W(level)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The programmed value is always a value without sign (absolute)

**Description**

Defines the maximum value for the position error in engineering units.  
 If the position error becomes larger in absolute value than the limit programmed with the function @40, the board automatically goes into emergency (the motor drives are disabled through the dedicated output).  
 If the limit is set at **zero**, the servocontrol mechanism is disabled (all the gains are disabled, no emergency limit) and the so-called **open loop** control is obtained.

e.g.: define the limit of the servocontrol

```
@40 X10 Y5 Z7
```

e.g.: read the limit of the servocontrol

```
%40 ZXY           ;request to read the limit of the servocontrol
X10 Y5 Z 7        ;reply from controller
```

## @41 / %41 Define Dead Band

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@41 [X(level)] [Y(level)] [Z(level)] [W(level)] X(level),Y(level),Z(level),W(level): dead band; range 0..99	(1) (2)
% syntax	%41 [X] [Y] [Z] [W] X,Y,Z,W: : dead band; range 0..99	(1)
CNC reply	[X(level)] [Y(level)] [Z(level)] [W(level)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The programmed value is always a value without sign (absolute)

**Description**

Defines, for each axis, the error limit of the position error in engineering units beyond which the servocontrol mechanism begins to act.

Normally **zero**, but, programming a value different from 0 in parameter @41 ensures that the servomechanism does not act until the position error exceeds, in absolute value, the set value.

e.g.: define dead band of the servocontrol

@41 X0 Y2

e.g.: read dead band of the servocontrol

```
%41 XY           ;request to read the dead band
X0 Y2           ;reply from controller
```



## @42 / %42 Define 2nd Servo Proportional Gain

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@42 [X(g)] [Y(g)] [Z(g)] [W(g)] X(g),Y(g),Z(g),W(g): proportional gain; range 0..10000	(1) (2)
% syntax	%42 [X] [Y] [Z] [W] X,Y,Z,W: : proportional gain; range 0..10000	(1)
CNC reply	[X(g)] [Y(g)] [Z(g)] [W(g)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) Programming **0** disables the proportional servo

**Description**

@42 can be used to define the value of the 2nd proportional gain (Kp2), for each axis..  
The correction made by the servocontrol mechanism will be, for the same position error, larger if the gain is higher.

$$\text{Correction} = Kp2 * E$$

where:

Kp2 = proportional gain

E = position error

Normally only one value of the proportional gain (@42) is used. Nevertheless in some applications, where a position error equivalent to some encoder pulse needs to be eliminated, it is possible to define 2 different values of proportional gain (@42 and @49), which intervene alternatively depending whether the absolute position error is greater (@42) or lesser (@49) than the programmed limit (@48).

e.g.: define 2nd servo proportional gain

@42 X12 Y20

e.g.: read 2nd servo proportional gain

%42 XY ;request to read\_2nd servo proportional gain  
X12 Y20 ;reply from controller

## @43 / %43 Define Servo Integral Gain

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@43 [X(g)] [Y(g)] [Z(g)] [W(g)] X(g),Y(g),Z(g),W(g): integral gain; range 0..10000	(1) (2)
% syntax	%43 [X] [Y] [Z] [W] X,Y,Z,W: : integral gain; range 0..10000	(1)
CNC reply	[X(g)] [Y(g)] [Z(g)] [W(g)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) Programming **0** disabled the servo integral

**Description**

@43 can be used to define the value of the integral gain (Ki) for each axis. It can be used to eliminate position errors due to the offset between the board and the motor drive. It affects the dynamic properties of the system (overshoot, oscillation, etc.) The applied correction, for the same error, increases in time with the programmed gain This correction can nevertheless be limited to a maximum value expressed in steps (see @47).

e.g.: define servo integral gain

@43 X12 Y20

e.g.: read servo integral gain

```
%43 XY           ;request to read servo integral gain
X12 Y 20         ;reply from controller
```

## @44 / %44 Define Servo Derivative Gain

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@44 [X(g)] [Y(g)] [Z(g)] [W(g)] X(g),Y(g),Z(g),W(g): derivative gain; range 0..10000	(1) (2)
% syntax	%44 [X] [Y] [Z] [W] X,Y,Z,W: : derivative gain; range 0..10000	(1)
CNC reply	[X(g)] [Y(g)] [Z(g)] [W(g)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) programming **0** disables the servo derivative

**Description**

@44 can be used to define the value of the derivative gain (Kd), for each axis.  
The correction of the velocity is a function of the programmed gain (@44) and it is connected to the variation in time of the position error (speed of variation of the error).  
For this reason, it is necessary to define the sampling time of the position error (see @45)

$$t = \text{programmed value (ms)}$$

e.g.: define derivative gain

@44 X12 Y20

e.g.: read derivative gain

```
%44 XY           ;request to read derivative gain
X12 Y 20         ;reply from controller
```

**@45 / %45 Define Derivative Sampling Time of the Servo**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@45 [X(d)[,(i)[,(p)]]] [Y(d)[,(i)[,(p)]]] [Z(d)[,(i)[,(p)]]] [W(d)[,(i)[,(p)]]] X(d,i,p),Y(d,i,p),Z(d,i,p),W(d,i,p): sampling time; range 1..9999	(1) (2)
% syntax	%45 [X] [Y] [Z] [W] X,Y,Z,W: : sampling time; range 1..9999	(1)
CNC reply	[X(d,i,p)] [Y(d,i,p)] [Z(d,i,p)] [W(d,i,p)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the value is expressed in thousandths of a second :  $T = n * 1 \text{ ms}$

**Description**

@45 can be used to define the sampling time (frequency of intervention) of the servo derivative, integral and proportional for each axis.

The correction of the speed is a function of the programmed gain (see @44) and it is connected with the variation in time of the position error (speed of variation of the error).

For this reason, the sampling time of the position error (@45) must be defined according to the formula  
 $t = 1 * \text{programmed value (ms)}$

For the most applications  $t=1\text{ms}$

e.g.: define sampling time

```
@45 X10,5,3 Y10,5,3 Z1,0,1
```

e.g.: read sampling time

```
%45 XYZ ;request to read sampling time
X10,5,3 Y10,5,3 Z1,0,1 ;reply from controller
```

## @46 / %46 Define Feedforward Percentage

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@46 [X(n)] [Y(n)] [Z(n)] [W(n)] X(n),Y(n),Z(n),W(n): feedforward percentage; range 0..100.0	(1) (2)
% syntax	%46 [X] [Y] [Z] [W] X,Y,Z,W: : feedforward percentage; range 0..100.0	(1)
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

Notes: (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

@46 can be used to define the percentage of the feedforward action to apply for each axis. The **theoretical velocity** is calculated automatically by the board in steps of velocity (the velocity value in open loop) and this can be applied as a percentage according to this parameter. The calculated value of the velocity will be:

$$\text{velocity} = \frac{\text{Theoretical velocity (@46)}}{100} * \text{programmed value}$$

Programming a value less than (example: 50 = 50% of the theoretical value) a proportion of the velocity value that the board has calculated is applied. This results in a greater intervention by the servocontrol mechanism.

e.g.: define feedforward percentage

@46 X1 Y1 Z50 ; set the percentage of application of the feedforward action  
; equal to 1% for X and Y , equal to 50% for Z

e.g.: read feedforward percentage

%46 XYZ ;request to read feedforward percentage  
X1.0 Y1.0 Z50.0 ;reply from controller

## @47 / %47 Define Servo Integral Limit

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@47 [X(n)] [Y(n)] [Z(n)] [W(n)] X(n),Y(n),Z(n),W(n): servo integral limit; range 0..32767	(1) (2)
% syntax	%47 [X] [Y] [Z] [W] X,Y,Z,W: : integral gain; range 0..32767	(1)
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the programmed value defines the limit, expressed in DAC steps, of the servo integral action  
programming **0** there are no limits set on the servo integral.

**Description**

@47 can be used to limit the servo integral action for each axis handled by the controller. The calculated velocity correction, for a given error, increases in time according to the programmed gain (see @43). This correction can be limited to a maximum value expressed in steps (@47).

e.g.: define servo integral limit

@47 X20 Y10 Z0

e.g.: read servo integral limit

```
%47 XYZ           ;request to read servo integral limit
X20 Y10 Z0       ;reply from controller
```

**@48/%48 Define Level between the 1st and 2nd Proportional Gain**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@48 [X(level)] [Y(level)] [Z(level)] [W(level)] X(level),Y(level),Z(level),W(level): level;	(1) (2)
% syntax	%48 [X] [Y] [Z] [W] X,Y,Z,W: : level	(1)
CNC reply	[X(level)] [Y(level)] [Z(level)] [W(level)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the programmed value is always a value without sign (absolute),  
if set at **0** the 1st proportional gain is disabled

**Description**

@48 is used to define the level at which the 2<sup>nd</sup> proportional gain intervenes for each axis. If the position error is less than the level programmed with @48, the 1st proportional gain is used (see @49). If it is above, the 2nd proportional gain (see @42) is used.

e.g.: define level

@48 X10 Y5

e.g.: read level

%48 XY	;request to read the level
X10.0 Y5.0	;reply from controller

**@49 / %49 Define 1° Servo proportional gain**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@49 [X(g)] [Y(g)] [Z(g)] [W(g)] X(g),Y(g),Z(g),W(g): proportional gain; range 0..10000	(1) (2)
% syntax	%49 [X] [Y] [Z] [W] X,Y,Z,W: : proportional gain; range 0..10000	(1)
CNC reply	[X(g)] [Y(g)] [Z(g)] [W(g)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) programming **0** disables the 1st servo proportional gain

**Description**

@49 is used to set the value of the 1<sup>st</sup> proportional gain (Kp1), in other words the proportional action of the servo that is used when the position error is between 0 and the level programmed with @48, for each axis.

The correction made by the servocontrol mechanism will be, for a given position error, greater for a higher programmed value of gain.

$$\text{Correction} = Kp1 * E$$

where:

Kp1 = proportional gain

E = position error

Normally a single value of proportional gain is used (see @42). Nevertheless, in some applications where the system response must be very rigid to maintain the axis in position, it will be necessary to set a high value for the gain. However, this should not influence the stability of the axis when moving. It is useful to set a high gain (Kp1; @49) that is valid only for small errors (level of a @48), while for larger errors, a lower gain is used (Kp2; @42).

e.g.: define 1st servo proportional gain

@49 X16 Y48

e.g.: read 1st servo proportional gain

%49 XY ;request to read of the 1st proportional gain  
X16 Y48 ;reply from controller



## @50 / %50 Define Space-Pulse Ratio of the Transducer

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@50 [X(s),(i)] [Y(s),(i)] [Z(s),(i)] [W(s),(i)] X(s),(i) Y(s),(i) Z(s),(i) W(s),(i): space and pulses	(1) (2)
% syntax	%50 [X] [Y] [Z] [W] X,Y,Z,W: : space and pulses	(1)
CNC reply	[X(s),(i)] [Y(s),(i)] [Z(s),(i)] [W(s),(i)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The programmed values are both without sign (absolute)

**Description**

This function is used to define the ratio between the displacement along the axis and the number of pulses from the transducer, for each axis, by defining two values (s = space, i = pulses).

The values (s) and (i), separated by the comma, must be greater than 0. The space can be entered in decimals (real number) while the number of pulses must be an integer

**Important**

In calculating the number of pulses, remember that the controller always multiplies the number of pulses received from the transducer by 4.

This parameter must be defined for all axes fitted with a position transducer.

e.g.: define space-pulses ratio of the transducer

```
@50 X6.5,100 Y6.5,100 Z10,1000
```

e.g.: read space-pulses ratio of the transducer

```
%50 XYZ ;request to read space-pulses ratio
X6.5,100 Y6.5,100 Z10,1000 ;reply from controller
```

\*\*\*\*

**@51 / %51 Define Space-Step Ratio of the Motor**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@51 [X(s),(p)] [Y(s),(p)] [Z(s),(p)] [W(s),(p)] X(s),(p) Y(s),(p) Z(s),(p) W(s),(p): spaces and steps	(1) (2)
% syntax	%51 [X] [Y] [Z] [W] X,Y,Z,W: : spaces and steps	(1)
CNC reply	[X(s),(p)] [Y(s),(p)] [Z(s),(p)] [W(s),(p)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The programmed values are both without sign (absolute)

**Description**

This function can be used to set the ratio between the displacement along the axis and the number of steps of the motor, for each axis, by entering two values

(s = space, p = step).

The values (s) and (p), separated by the comma, must be greater than 0. The space can be entered in decimals (real number) but the number of pulses must be an integer.

This instruction is only meaningful for axes driven by a stepper motor.

**Important**

In calculating the number of steps, take into account the way the motor works (whole step, half step, quarter step, etc. ).

e.g.: define space-step ratio of the motor

@51 X10.234,100 Y0.5,10 Z10.99,1000

e.g.: read space-step ratio of the motor

%50 XYZ	;request to read space-step ratio
X10.234,100 Y0.5,10 Z10.99,1000	;reply from controller

## @52 / %52 Define Analogue voltage ratio

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@52 (i)[(g2),(v2) [(g1),(v1)]] (i) : identifier (g1) : r.p.m. or velocity of first point (default = 0) (v1) : associated voltage of first point ( default = 0) (g2) : r.p.m. or velocity of second point (v2) : associated voltage of second point	
% syntax	%52	
CNC reply	(i),(g2),(v2) [(g1),(v1)]	

**Description**

This function can be used to define the ratio between the velocity of an axis or the r.p.m. of a spindle (g) and the voltage obtained at the output.

The identifying character (i) can have the following meanings:

- ' ' => disabled (default)
- 'S' => spindle (analogic output enables from the instruction ISO S)
- 'X' => X-axis (analogic output proportional of the axis X velocity)
- 'Y' => Y-axis (analogic output proportional of the axis Y velocity)
- 'Z' => Z-axis (analogic output proportional of the axis Z velocity)
- 'W' => W-axis W (analogic output proportional of the axis W velocity)
- 'I' => point velocity in interpolation (analogic output proportional of the interpolation velocity)

Setting only one point defines a characteristic that passes the zero (rotation velocity or movement equal to zero has an output voltage of zero volts).

Setting both points (different from each other) the characteristics are obtained that do not pass through the zero (velocity equals zero when the output voltage is different from zero).

**note (1):** The addresses Y, Z and W are allowed only if available on the chosen model of controller

e.g.: define analogue voltage ratio

```
@52 S8000,10           ;with the instruction S800, there is an output voltage of 1 Volt
@52 I0,-1.5 6000,0    ;in the interpolated movements with stationary axes, there is an output
                       ;voltage of -1.5 Volt and a voltage of 0 Volt at a speed F6000
```

e.g.: read analogue voltage ratio

```
%52                   ;request to read analogue voltage ratio
S8000,10              ;reply from controller
%52                   ;request to read analogue voltage ratio
I6000,5 1000,-2      ;reply from controller
```

**Note**

Obviously the instruction is meaningful only if the optional analogue output is present.

## F@53 / %53 Define Maximum Frequency

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@53 [X(f)] [Y(f)] [Z(f)] [W(f)] X(f), Y(f), Z(f), W(f) :maximum frequency	(1) (2)
% syntax	%53 [X] [Y] [Z] [W] X,Y,Z,W: : maximum frequency	(1)
CNC reply	[X(f)] [Y(f)] [Z(f)] [W(f)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the programmed values are expressed in Hz (Hertz)

**Description**

@53 is used to define the values of maximum frequency of each axis.  
 If the axis is d.c. driven, the value defines the maximum frequency of the output from the transducer (pulses per second) when the axis is moving at the maximum velocity.  
 If the axis is driven by a stepper motor (with or without position feedback) the value defines the maximum frequency that the controller can generate.

**Important**

In calculating the values of frequency, remember that the controller always multiplies the pulses received from the transducer by 4.  
 Consider also how the stepper motor works (whole step, 1/2 step, 1/ 4 step, etc. ).

If the instruction @84,1 ha been programmed and also the instruction @54 for the same axis is programmed =0, the parameter (@53) is used as a limit of velocity.

e.g.: define maximum frequency

@53 X15347 Y18000 Z1000

e.g.: read maximum frequency

%53 XYZ	;request to read maximum frequency
X15347 Y18000 Z1000	;reply from controller

## @54 / %54 Programming Velocity Limit

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@54 [X(v)] [Y(v)] [Z(v)] [W(v)] X(v), Y(v), Z(v), W(v) :velocity limit	(1) (2)
% syntax	%54 [X] [Y] [Z] [W] X,Y,Z,W: : velocity limit	(1)
CNC reply	[X(v)] [Y(v)] [Z(v)] [W(v)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the value represents a velocity, expressed in engineering units / min

**Description**

The function @54 is used to define the values of maximum velocity of each axis.  
The instruction is enabled with @84.  
With this instruction, it is possible to define a velocity limit for each axis as a function of the mechanical properties of the axis.

**Important**

If the velocity limit programmed is equal to 0 (thus the default value), the value used is taken from @53.

e.g.: define velocity limit

```
@54 X15000 Y20000 W10000
```

e.g.: read velocity limit

```
%53 XYW ;request to read velocity limit
X15000 Y20000 W10000 ;reply from controller
```

**@56 / %56 Define Velocity / Frequency of Start-Stop**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@56 [X(ss)] [Y(ss)] [Z(ss)] [W(ss)] X(ss), Y(ss), Z(ss), W(ss) :frequency / velocity start-stop; range 0..9999	(1) (2)
% syntax	%56 [X] [Y] [Z] [W] X,Y,Z,W: : frequency / velocity start-stop; range 0..9999	(1)
CNC reply	[X(ss)] [Y(ss)] [Z(ss)] [W(ss)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) For d.c. motors the value represents a velocity expressed in engineering units/min. For stepper motors it represents a frequency expressed in Hz.

**Description**

The function @56 is used to define the values of the start-stop velocities for the axes with d.c.motors and the start-stop frequencies for axes with stepper motors.  
The values of frequency for the axes with stepper motors must always be integers.

**Important**

In calculating the values of frequency, always take into account how the stepper motor works (whole step, 1/2 step, 1/4 step, etc. ).

e.g.: define start-stop velocity / frequency

@56 X0 Y1.5 Z100 ;The X- and Y-axes have d.c. motors, the Z-axis has a stepper motor

e.g.: read start-stop velocity / frequency

%56 XYZ ;request to read start-stop velocity / frequency  
X0 Y0 Z100 ;reply from controller : The X- and Y-axes have d.c. motors, the Z-axis has a stepper motor

**@57 / %57 Define Acceleration/Deceleration Ramp**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@57 [X(t)] [Y(t)] [Z(t)] [W(t)] X(t), Y(t), Z(t), W(t) : ramp time; range 0..9999	(1) (2)
% syntax	%57 [X] [Y] [Z] [W] X,Y,Z,W: : ramp time; range 0..9999	(1)
CNC reply	[X(t)] [Y(t)] [Z(t)] [W(t)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the value is expressed in thousandths of a second  
programming 0 produces a velocity step

**Description**

The function @57 is used to define the gradient of the acceleration and deceleration ramps for each axis, using the programming of the time to take from the start-stop velocity or frequency to the maximum velocity or frequency (defined by code @53) and viceversa.

e.g.: define acceleration/deceleration ramp

@57 X1000 Y500 Z800

e.g.: read acceleration/deceleration ramp

%57 Z Y X ;request to read acceleration/deceleration ramp  
X1000 Y500 Z800 ;reply from controller

\*\*\*\*

**@59 / %59 Define Level per Axis in Position**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@59 [X(level)] [Y(level)] [Z(level)] [W(level)] X(level), Y(level), Z(level), W(level) : level	(1) (2)
% syntax	%59 [X] [Y] [Z] [W] X,Y,Z,W: : level	(1)
CNC reply	[X(level)] [Y(level)] [Z(level)] [W(level)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) the programmed value is expressed in encoder pulses

**Description**

@59 is used to define the level, for each axis, of the position error, under which the axis is considered to be "in position".

On this subject, see also the functions G61 and G62.

e.g.: define level for axis in position

@59 X5 Y2 Z0

e.g.: read level for axis in position

%59 X Y Z	;request to read level
X5 Y2 Z0	;reply from controller



**@60-@61 / %60-%61 Define Machine Limits**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@60 [X(co-ord.)] [Y(co-ord.)] [Z(co-ord.)] [W(co-ord.)] X(co-ord.), Y(co-ord.), Z(co-ord.), W(co-ord.) : limits	(1) (2)
% syntax	%60 [X] [Y] [Z] [W] X,Y,Z,W: : limits	(1)
CNC reply	[X(co-ord.)] [Y(co-ord.)] [Z(co-ord.)] [W(co-ord.)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The limit co-ordinates can be programmed with sign and are always referred to the machine zero

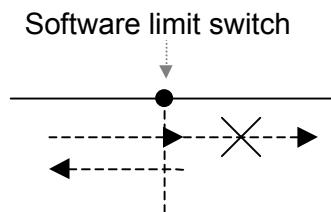
**Description**

These two functions are used to define the limit values of the co-ordinates of each axis, referred to the machine zero..

@60 is used to define the maximum limits, and @61 for the minimum limits.

The values programmed with these functions can be reduced further during operation with the functions G25 and G26.

If the software limit switch is exceeded, the controller goes into emergency and the motor drive is disabled. Once the controller is reset, it will move only away from the limit switch (see figure). If it tries to move beyond the limit switch again, the controller will go into emergency again.



**Important**

If both limits of an axis are set to 0, there will be no monitoring of exceeding the co-ordinates. This is useful for rotary axes where there are no physical limits.

**Note bene**

- The software limit switches, if programmed, are not active during the SET-POINT phase.
- The value defined by @60 must be greater than that defined by @61. If this is not so, the controller will go into emergency as soon as it starts to execute any instruction of movement.

e.g.: define limits

@60 X100.123 Y500.000 Z100  
@61 X-100 Y-500.000 Z0

e.g.: read limits

%60 XYZ ;request to read limits  
X100.123 Y500.000 Z100 ;reply from controller

%61 XYZ ;request to read limits  
X-100.000 Y-500.000 Z0 ;reply from controller

**@70 / %70 Define Panel key active states**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@70 K(t),(b) [K(t),b] [K(t),(b)] .... [K(t),(b)] K(t),(b) : number of key, type of key	(1)
% syntax	%70	
CNC reply	[K(t),b] [K(t),(b)] .... [K(t),(b)]	

Notes: (1) the value (t) must be in the range 1-16 (a maximum of 16 keys may be defined)  
the value (b) must be in the range 0-17

**Description**

The function @70 is used to define the active states of the keys associated with the external control panel.

The “operator panel” is enabled by the instructions :  
 @97,1 ; enable the “operator panel” on power-up  
 &10 - &11 ; to enable / disable the “operator panel” on-line

The value (b) has the following meanings:

- b = 0 -> key disabled
- b = 1 -> active only when pressed in JOG
- b = 2 -> active only when pressed in AUTOMATIC
- b = 3 -> active only when pressed in AUTOMATIC and in JOG
- b = 4 -> key disabled
- b = 5 -> active both when pressed and when released in JOG
- b = 6 -> active both when pressed and when released in AUTOMATIC
- b = 7 -> active both when pressed and when released in AUTOMATIC and in JOG
- b = 8 -> select AUTOMATIC / JOG mode;
- [ b = 9 -> select PROG / LINE mode] ;
- b = 10 -> select program (bit 0) ; instructions for handling
- b = 11 -> select program (bit 1) ; the “operator panel”
- b = 12 -> select program (bit 2) ; without the help of the PLC
- b = 13 -> select program (bit 3) ;
- b = 14 -> select program (bit 4) ; the inputs of the unused program
- b = 15 -> select program (bit 5) ; selectors are set to = 0
- b = 16 -> select program (bit 6) ;
- b = 17 -> select program (bit 7) ;

each key can have two “programs” associated with it, one when it is pressed and another when it is released. The association of programs to the key numbers is the following:

- key 1 (K1) -> program 101 when pressed  
program 121 when released
- key 2 (K2) -> program 102 when pressed  
program 122 when released

key 16 (K16) <sup>.....</sup> -> program 116 when pressed  
<sup>.....</sup>  
program 136 when released

e.g.: define panel key active states

@70 K1,2 K2,0 K3,5 ; key 1 is defined as only when pressed in automatic (program 101)  
; key 2 is disactivated  
; key 3 is defined as active when pressed and when released in manual  
(programs 103 and 123);

e.g.: read panel key active states

%70 request to read active key states  
K1,2 K3,5 *reply from controller*

**@71 / %71 Define Data Format Sent by the Controller**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@71 [X(d)] [Y(d)] [Z(d)] [W(d)] X(d), Y(d), Z(d), W(d) : decimals ; range 0..5	(1)
% syntax	%71 [X] [Y] [Z] [W] X,Y,Z,W: : decimals; range 0..5	(1)
CNC reply	[X(d)] [Y(d)] [Z(d)] [W(d)]	

Notes: (1) The programming concerns only the inputs of the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller

**Description**

The function @71 is used to define the format of the data sent by the controller in reply to any instruction of read co-ordinate, space and velocity, programming the number of decimal digits of the value sent.

e.g.: define data format

@71 X1 Y1 Z3

e.g.: read data format

%71 X Y Z ;request to read data format  
X1 Y1 Z3 ;reply from controller

**@72 / %72 Define Type of Axis: Linear / Angular**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@72 [X(n)] [Y(n)] [Z(n)] [W(n)] X(n), Y(n), Z(n), W(n) : type of axis	(1) (2)
% syntax	%72 [X] [Y] [Z] [W] X,Y,Z,W: : type of axis	(1)
CNC reply	[X(n)] [Y(n)] [Z(n)] [W(n)]	

- Notes:
- (1) The programming concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the chosen model of controller
  - (2) The character (n) can have the following values:  
0 = linear;      1 = angular

**Description**

The function is for defining whether the axis is linear or angular.  
On the co-ordinates of the angular axes, the conversion to inches is never applied.

e.g.: define type of axis

@72 X0 Y0 Z1

e.g.: read type of axis

%72 X Y Z  
X0 Y0 Z1

;request to read type of axis  
;reply from controller

**@80 / %80 Enable the Percentage Variation of the Axes' Velocity**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@80,(n) (n) : 0 = disable; 1 = enable	(1)
% syntax	%80	
CNC reply	(n)	

Notes: (1) No address is needed  
The value (n) is separated from the function code by a comma

**Description**

The function is used to enable or otherwise the percentage correction of the nominal velocity attributed to the axes.

e.g.: enable percentage variation

@80,1

e.g.: read if percentage variation is enabled

%80 ;request to read if percentage variation is enabled  
1 ;reply from controller

<b>%81 Read Variables</b>
---------------------------

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
% syntax	%81 (variable)	<b>(1)</b>
CNC reply	“value of the variable”	

Notes: (1) The name of the variable to read must be specified

**Description**

The function is used to read the contents of the variables (1-255). It (variable) can be expressed as a number (1..255) preceded by the letter Q (e.g. Q120 for the variable 120) or the mnemonic that represents this variable (e.g. QRX for the variable 219).

e.g.: read variables

```
%81 Q21           ;request to read variables
24234.5          ;reply from controller
```



<b>@82 / %82 Reserved Instruction</b>
---------------------------------------

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@82,(n)	
% syntax	%82	
CNC reply	(n)	

<b>@83 / %83 Enable Line in Execution</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@83,(n) (n) : 0 = disabled; 1 = enabled	<b>(1)</b>
line in execution enable/disable CNC reply	(n)	

Notes: (1) No address is required  
The value (n) is separated from the function code by a comma

**Description**

The function is used to enable or disable the sending, at the end of the number of the line in execution, the line of program.

e.g.: enable line in execution

```
@83,1
ACK
N15 G0 X100
```

e.g.: read whether line in execution is enabled

```
%83          ;request to read whether line in execution is enabled
1            ;reply from controller
```

## @84 / %84 Enable Velocity Limit

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@84,(n) (n) : 0 = disabled; 1 = enabled	<b>(1)</b>
% syntax	%84	
CNC reply	(n)	

Notes: (1) No address is required  
 The value (n) is separated from the function code by a comma  
 The default value is '0'

**Description**

The function enables the parameter programmed with the @53 or @54 instruction as a maximum limit of motor velocity.

If the value is set to '0', the controller does not use this value as the maximum limit and can go beyond it.

If a value greater than zero has been set for the function @54, this value is used; otherwise the value entered with @53 is used.

e.g.: enable maximum limit

@84,1

e.g.: read whether maximum limit is enabled

```
%84          ;request to read whether maximum limit is enabled
1            ;reply from controller
```

**@85 / %85 Enable <BEL> Code Transmission**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@85,(n) (n) in the range : 0..9	(1) (2)
% syntax	%85	
CNC reply	(n) : 0 = <BEL> disabled; 1..9 = timing expressed in seconds	

- Notes:
- (1) No address is required address  
The value (n) is separated from the function code by a comma
  - (2) (n) defines the time, expressed in seconds, that must elapse between the transmission of two successive <BEL> codes  
programming **0** disables the transmission

**Description**

The function is used to enable or disable the timed transmission of the instruction code Ctrl-G (<BEL>) during a state of emergency or of alarm of the controller.

e.g.: enable <BEL> code transmission

@85,4

e.g.: read whether <BEL> code transmission is enabled

%85 ;request to read  
4 ;reply from controller

**@86 / %86 Modify the Communication Protocol (PLC)**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
@ syntax	@86 , (t) (t) : number in the range 0..23	
% syntax	%86 (n)	

CNC reply (n)

**Description**

This instruction is used to modify the protocol of communication between the PC and the controller. On power-up, the system is set up to communicate with the following serial protocol :

```

baud-rate      = 9600
data-bits      = 8
parity         = ODD
stop-bits      = 1
    
```

With this instruction it is possible to modify the BAUD-RATE and the PARITY . The parameter (t) is a number (0-23) with the following values :

$t = bd + prt * 8$       in which :

- bd =0 -> 300 baud
- bd =1 -> 600 baud
- bd =2 -> 1200 baud
- bd =3 -> 2400 baud
- bd =4 -> 4800 baud
- bd =5 -> 9600 baud
- bd =6 -> 19200 baud
- bd =7 -> 38400 baud

- prt =0 -> EVEN
- prt =1 -> ODD
- prt =2 -> NO PARITY

e.g.: modify the communication protocol PLC

```

PC          > '@86,22' <cr>      select the protocol 19200,8,1,NO-PARITY
CONTROLLER > '$'
    
```

e.g.: lecture the communication protocol PLC

```

PC          > '%86' <cr>
CONTROLLER > '22'
    
```

**@90 / %90 Define Error Level of Co-ordinates of the Centre**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@90,(level) (level) in the range : 0..99.9	(1) (2)
% syntax	%90  CNC reply (level)	

- Notes:
- (1) No address is required  
the error level is separated from the function code by a comma
  - (2) the programmed value is always a value without sign (absolute) 0...99.9

**Description**

The function is used to define the maximum error that can be allowed in the program when calculating the co-ordinates of the centre in the circular interpolation.

e.g.: define level

@90,0.5

e.g.: read level

%90	;request to read
0.5	;reply from controller

**@91 / %91 Define “Objective Reached” Level**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@91,(level) (level) in the range : 0.0..qmax	(1) (2)
% syntax	%91  CNC reply (level)	

- Notes:
- (1) No address is required  
the level is separated from the function code by a comma
  - (2) the programmed value is always a value without sign (absolute)

\*\*\*\*

**Description**

The function is used to define the value of the position error below which the bit of “objective reached” is activated.

e.g.: define level

@91,20.0

e.g.: read level

%91	;request to read
20.0	;reply from controller

**@94 , %94 Define Tangential Tool "LEVEL"**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@94,(s) (s): define "level" for "TANGENTIAL" tool The value (s) must be an absolute value without sign	<b>(1)</b>
% syntax	%94	<b>(1)</b>
CNC reply	(s)	

Notes: (1) The programming concerns the axis defined as "TANGENTIAL AXIS "

**Description**

The value of this function defines a position window inside which the tangential tool must be found (axis programmed as TANGENTIAL) in order that, a subsequent movement in linear or circular interpolation of the two axes in the active plane of interpolation can proceed without performing an operation of:

RAISE, POSITION, LOWER tool.

As a function of the "level" programmed, the controller handles the operations of positioning the tool when the automatic guide is active (see instruction G93).

Suppose it is working in the X-Y plane, with the Z-axis being TANGENTIAL and to have, because of previous movements, positioned the Z-axis at the co-ordinate 44.5 degrees and to require to execute a movement of interpolation on the XY axes such that the Z-axis should be raised, positioned at 45 degrees and lowered before beginning the interpolation.

Programming a level of 1 degree using @94, a window between 44 and 46 degrees is defined. The positioning of the tool will take place without performing the operation of raising and lowering and will be contemporaneous to the execution of the movement of interpolation.

Following this description, it is advised not to use this level in applications such as a plotted cut in which the accuracy and the mode of the positioning, which must always take place when the tool is raised, is fundamental to the quality of the work being performed.

e.g.: programming tangential tool "level"

@94,1.5 ;define tangential tool "level" = 1.5

e.g.: read tangential tool "level"

%94 ;request to read  
3.5 ;reply from controller ("level" = 3.5)

**N.B.** The data of the programmed, and/or read, "level" is always expressed in the units of measurement defined for the axis

Programmed as "TANGENTIAL" tool instructions: G93 , @71 ).

**Note**

Instruction is available if the controller is installed with the relative option.



**@95 , %95 Define " NO. OF CUTTING EDGES " of the tangential tool**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@95,(n) (n): define NO. of CUTTING EDGES of the tangential tool (allowed values 1 or 2) The value (n) can have the following values: 1 = tangential tool with "1 CUTTING EDGE 1" 2 = tangential tool ad "2 CUTTING EDGES"	<b>(1)</b>
% syntax	%95	<b>(1)</b>
CNC reply	(n)	

Notes: (1) The programming concerns the axis defined as the "TANGENTIAL AXIS"

**Description**

The handling is always associated with the axis programmed as **TANGENTIAL** (rotary axis), and it enables the NUMBER OF CUTTING EDGES (1 or 2) of the tool to be defined.  
 As a function of the no. of cutting edges, the positioning operations of the tool are optimised when the automatic guide is active (see instruction **G93**).

e.g.: programming tangential tool with 2 cutting edges

@95,2 ;defines tangential tool = tool with 2 cutting edges

e.g.: read type tangential tool

%95 ;request to read

1 ;reply from controller (tangential tool with 1 cutting edge)

**Note**

Instruction available if the controller is installed with the relative option

**@96 / %96 Define tangential tool "RAISE/LOWER TIMES"**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@96,(a),(b) (a),(b): define RAISE/LOWER TIMES OF "TANGENTIAL" tool The values (a),(b) can have the values: (a) = "RAISE TIME" (range: 0...9999 milliseconds) (b) = "LOWER TIME" (range: 0...9999 milliseconds)	(1)
% syntax	%96	(1)
CNC reply	(a),(b)	

Notes: (1) The programming concerns the axis defined as the "TANGENTIAL AXIS"

**Description**

The handling is always associated with the axis programmed as **TANGENTIAL** (rotary axis) and it enables a minimum time to be set for the interval between the raising of the tool and its rotation (raise) and the minimum time between the lowering of the tool and the re-starting of the other two axes on the plane of interpolation.

In handling the RAISE/LOWER of the tool, it is also possible to subject the relative movements to specific inputs (TOOL-LOW and TOOL-HIGH) programmed with instruction @01; the execution sequence of a cycle of RAISE/LOWER, in this condition, becomes :

- Activate the "tool up" output
- Await activation of the "tool up" input
- Begin counting the delay time ("RAISE TIME")
- When the delay time has expired, activate the rotation of the tool
- Activate the output "tool down" output
- Await activation of the "tool down" input
- Begin counting the delay time ("LOWER TIME")
- When the delay time has expired, re-start the movement of the axes in the plane of interpolation

The two functions are mutually exclusive (the tool is either up or down), therefore if one of the two functions is handled by a limit switch, the other function, apart from waiting for the set time, verifies that the first function is no longer active.

e.g.: programming RAISE/LOWER times of tangential tool

```
@96,200,500           ;defines time of "RAISE TOOL" = 200ms
                       ;time of "LOWER TOOL" = 500ms
```

e.g.: read tangential tool type

```
%96                 ;request to read
1                   ;reply from controller (tangential tool with 1 cutting edge)
```

**Note**

Instruction available if the controller has the relative option installed.

**@97 / %97 Enable Control Panel on Power-up**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@97, (n) (n) : 0 = disabled; 1 = enabled	<b>(1)</b>
% syntax	%97	
CNC reply	(n)	

Notes: (1) No address is required  
 The value (n) is separated from the function code by a comma

**Description**

The function is used to define is the remote control panel must be enabled automatically.

e.g.: enable control panel

@97,1

e.g.: read if control panel enabled

%97 ;request to read  
 1 ;reply from controller

<b>@98 Recover Parameters from Non-volatile Memory</b>
--

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@98	<b>(1)</b>

Notes:                   (1) Is an instruction without parameters

**Description**

This function causes the “current” values of all the parameters definable with ‘@’ functions to be read in block from the non-volatile memory.

The values read are those saved with the instruction @99.

e.g.: recover the parameters from the battery backed-up RAM

@98

## %98 Read Configuration of the Control System

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
% syntax	%98	<b>(1)</b>
CNC reply	m a , x y z w and o i p [nn]	

Notes: (1) The function does not contain an address

**Description**

This function is for reading the configuration (prememorised in the controller) and to verify the following properties of the controller :

- model (positioner, interpolator or tangential tool)
- number of axes (1, 2,3 o 4)
- type of axes (d.c. motor, stepper motor or stepper motor with position feedback)
- number of modules of I/O in PLC
- number of modules of analogue outputs in PLC
- number of modules of analogue inputs in the PLC
- number of the controller (only if defined a number of control different from 0)

**Reply**

The reply is made up of a string of ASCII characters:

"m" "a" ',' "x" ["y"] ["z"] ["w"] "e" "o" "i" ["(nn)"]

where :

"m" is a character that indicates the model :

- 'P' : positioner
- 'I' : interpolator
- 'U' : tangential tool

"a" is a character that indicates the number of axes :

- '1' : 1 axis
- '2' : 2 axes
- '3' : 3 axes
- '4' : 4 axes

"x" "y" "z" and "w" are characters that indicate the type of axis :

'C' : d.c. motor

'P' : stepper motor

'R' : stepper motor with position feedback

'-' : not enabled

"e" is a character that indicates the number of available I/O modules :

"o" is a character that indicates the number of available analogue outputs:

"i" is a character that indicates the number of available analogue inputs:

"p" can assume the value:

"C": without PLC

" " : with PLC

"nn" is a number (between 01 and 99) that indicates the number associated with the controller.

e.g.: read configuration

```
%98           ;request to read  
I3,CCP110     ;reply from controller
```

<b>@99 Save Parameters in the Non-volatile Memory</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
@ syntax	@99	<b>(1)</b>

Notes: (1) Is a function without parameters

**Description**

This function causes the “current” values of all the parameters definable with the ‘@’ functions to be saved in block in the non-volatile memory.  
 The values saved can be recalled at any moment with the instruction @98.

e.g.: save parameters

@99

<b>%99 Read Software Version</b>
----------------------------------

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
% syntax	%99	<b>(1)</b>
CNC reply	“controller name” Vx.xx , CNCPLC Rel x.xx CNT Relx.xx	

Notes: (1) No address is required

**Description**

Consents the reading of an ASCII string that indicates the version of the software installed in the controller.

Reply of the Cnc:

“controller name” Vx.xx , CNCPLC Rel x.xx CNT Relx.xx





# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 5

-

“G”

**Instructions  
(Operative Instructions )**



## CONTENTS

CHAPTER 5 - “G” INSTRUCTIONS (OPERATIVE INSTRUCTIONS ).....	5-1
Introduction .....	5-4
G00 (G0) Positioning axes .....	5-5
G01 (G1) Linear interpolation .....	5-6
G02 (G2) Clockwise circular interpolation.....	5-7
G03 (G3) Anticlockwise Circular interpolation .....	5-10
G04 (G4) Hold at end of block.....	5-11
G06 (G6) Velocity controlled axis .....	5-12
G16 Define the Plane of Circular interpolation.....	5-13
G17 - G18 - G19 Select the Plane of Circular interpolation .....	5-14
G20 Unconditioned jump .....	5-15
G21 - G22 Conditioned jump.....	5-16
G25 - G26 Define Field of Work limits .....	5-17
G27 Cancel Field of Work limits .....	5-18
G30 Recall a Subprogram.....	5-19
G31 - G32 Conditioned recall of Subprogram.....	5-20
G50 Cancel Displacement of Origin of the Axes.....	5-21
G51 Seek origin of axes (set-point) .....	5-22
G52 Displacement of Axes Origin.....	5-23
G53 Origins of Axes at this Point.....	5-24
G54 Origins of Axes at this Point (software set-point).....	5-25
G55 – G56 Save – Restore Origin.....	5-26
G57 – G58 Activate - Disactivate Automatic Linkage.....	5-28
G61–G62 Activate-Disactivate “Accurate Stop (in position)” .....	5-30
G63 – G64 Not Wait (Wait) for End of Movement.....	5-31
G65-G66 Hold Input / Status Bit .....	5-32
G67-G68 Handling Devices.....	5-34
G69 Assign State to the “Flag” .....	5-36
G70 G71 Unit of Measurement.....	5-38
G74 Enabling and disabling the co-ordinate reading by laser .....	5-39
G80 Enable and Disable Cam Table and Engage Posn. ....	5-40
G81 Define DISENGAGE Positions of Cam .....	5-41
G82 Define Automatic Cam Table .....	5-42
G83 Define Cam K Factor .....	5-43
G84 Define Cam Velocity Variations in %.....	5-44
G90 G91 Programming Absolute or Incremental Co-ordinates .....	5-45
G93 Activate “Tangential Tool Guide”.....	5-46
G94 Disactivate “Tangential Tool Guide”.....	5-47
G98 Velocity handling in positioning and interpolation.....	5-48

## Introduction

The G codes are the main instructions used by the controller to perform the fundamental operations, such as:

- Positioning
- Linear interpolation
- Circular interpolation
- Tangential interpolation
- Seek and set the origins
- Control of the co-ordinates
- Inputs Test
- Activation of the outputs
- Wait

## Modal functions

Some G functions are defined as “modal” as when they are programmed once, they remain active until a different G function complementary to them (also modal) cancels the effect.

The “modal” G functions will be highlighted in the description of the individual instructions, indicating also the complementary function.

## Compatible functions

Some G codes can be used contemporaneously in the same program block, which is the definition of their “compatibility”.

In the description of each function, there is information regarding the syntax of the function, which indicates also the compatible functions.

The functions, in whose description compatible functions are not indicated, must be the only ones present in the program block.

# G00 (G0) Positioning axes

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

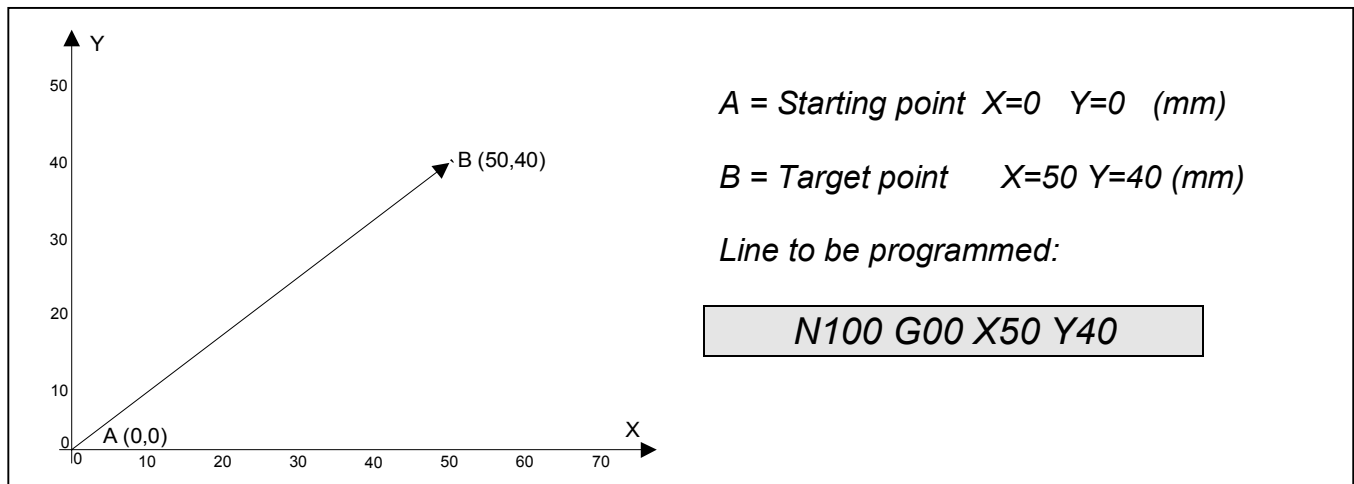
Source	Host computer / Resident program	
Active	Until the execution of instructions G01, G02, G03	<b>(1)</b>
Execution	Queued	
Conditions	-	
Compatible functions	G70, G71, G90, G91	
Syntax	G00 [Compatible G's] [X...] [Y...] [Z...] [W...] [F...] X,Y,Z,W: co-ordinate of the final point F: velocity of the axes	<b>(2)</b>

- Note:
- (1) Active automatically on power-up of the CNC
  - (2) The movement concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

Contemporaneously positions the axes specified in the instruction. The programmed position is reached independently by the axes with the velocity (address F) programmed in the block, or, if not mentioned, with the same velocity as the last movement programmed.  
The co-ordinates programmed can be absolute (G90) or incremental (G91) and expressed in mm. (G71) or in inches (G70).

e.g.: Positioning from A to B in the X-Y plane



Note (1) : The axes arrive independently at the co-ordinate specified by the instruction and therefore the trajectory followed could differ from that shown in the figure.

e.g.: Execution of a series of positionings

- |                          |   |
|--------------------------|---|
| G00 G90 G71 X100.5 F2000 | ;Position the X-axis at 100.5 mm with a velocity of 2000 mm/min               |
| G70 Y200                 | ;Position Y-axis at 200 inches with a speed of 2000 inches/min                |
| G71 X200 Y300 Z500 F5000 | ;Position X,Y,Z respectively at 200,300,500 mm with a velocity of 5000 mm/min |
| G0 X100 G91 Y-50         | ;Position X at 100 and Y at 250 mm (300-50) with a velocity of 5000 mm/min    |

# G01 (G1) Linear interpolation

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G00, G02, G03	
Execution	Queued	
Conditions	Not allowed in the "Positioner" models	
Compatible functions	G70, G71, G90, G91	
Syntax	G01 [Compatible G's] [X...] [Y...] [Z...] [W...] [F...] X,Y,Z,W: co-ordinate of the final point F: velocity of the point	<b>(1)</b>

Notes: (1) The movement concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the model of controller selected

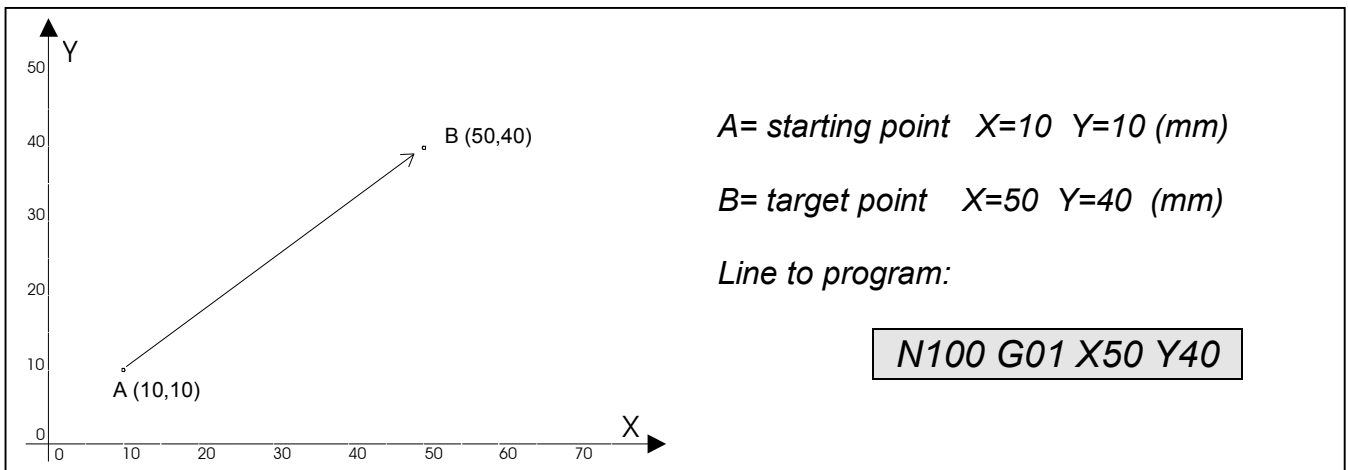
**Description**

Execute a movement in linear interpolation between the axes programmed in the block (X,Y,Z,W); the programmed position is reached with co-ordinated movements of the 2, 3 or 4 axes (the final point is reached by all the axes at the same time).

The velocity of the movement (F) is that programmed in the block or, if omitted, the last programmed velocity, and referred to the point on the trajectory (straight).

The co-ordinates of the final point can be absolute (G90) or incremental (G91) and expressed in mm (G71) or in inches (G70)

e.g.; Linear interpolation from A to B in the X-Y plane



e.g.; Execution of a series of linear interpolations

- G01 G90 X100 Y150 F2000 ;move the X- and Y-axes to 100 mm and 150 mm with a velocity of 2000 ;mm/min
- G70 Y200 ;move the Y-axis to 200 inches with a velocity of 2000 mm/min
- G71 X200 Y300 Z500 F5000 ;move X,Y and Z respectively to 200,300 and 500 mm with a velocity of 5000 mm/min
- G1 X100 G91 Y-50 ;move X to 100 and Y to 250 mm (300-50) with a velocity of 5000 mm/min

## G02 (G2) Clockwise circular interpolation

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

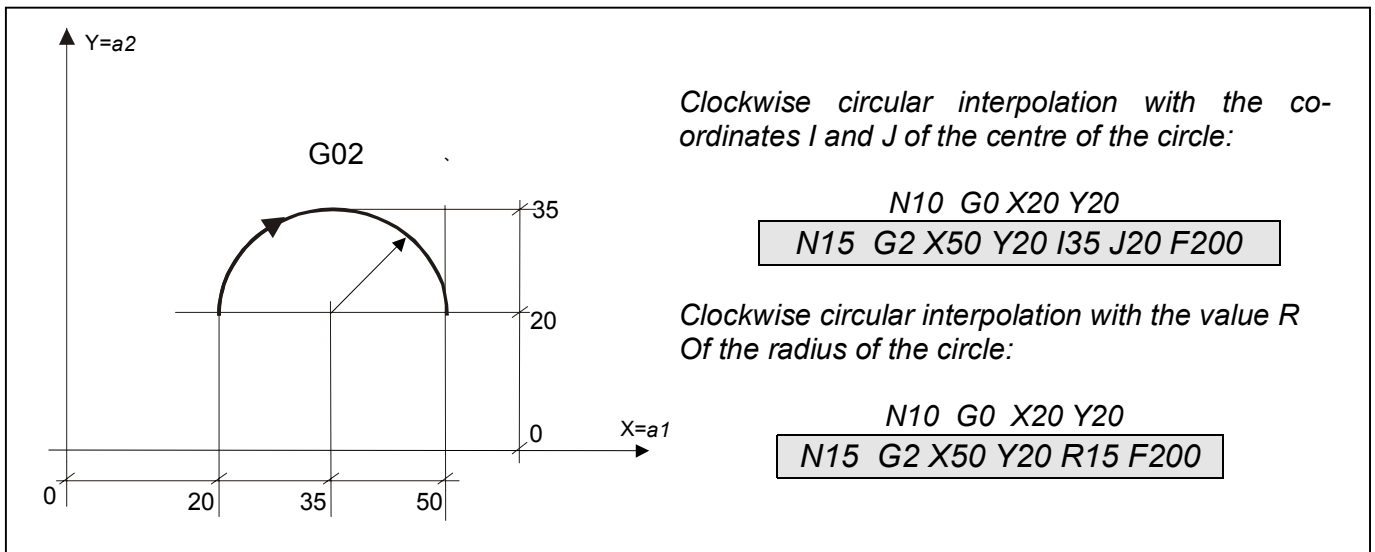
Source	Host computer / Resident program	
Active	Until the execution of instructions G00, G01, G03	
Execution	Queued	
Conditions	Not allowed in the "Positioner" models and in models with only one axis	
Compatible functions	G70, G71, G90, G91	
Syntax	G02 [Compatible G's] [X...] [Y...] [Z...] [W...] [I...,J...] [R...] [F...] X,Y,Z,W: co-ordinate of the final point I: co-ordinate of centre 1st axis (a1) J: co-ordinate of centre 2 <sup>nd</sup> axis (a2) R: radius F: velocity of the point	(1) (2) (2) (3)

- Notes:
- (1) The two axes to be programmed are those of the work plane selected (see functions G16,G17,G18,G19).  
The addresses Z and W are allowed only if available on the model of controller selected
  - (2) a1 and a2 are the axes defined with instruction G16
  - (3) The radius R is alternative to the co-ordinates I and J: it is not allowed to program both the radius and the centre of the circle contemporaneously.

**Description**

Execute a movement in circular interpolation in the clockwise direction in the plane selected previously with the functions G16,G17,G18 and G19. The programmed position is reached with a circular type of movement and co-ordinated on both axes (the target is reached contemporaneously by both axes). The velocity of the displacement (F) is that programmed in the instruction or, if it is not mentioned, the last velocity programmed, and it refers to the trajectory (arc of a circle) followed by the axes. The arc of the circle to follow can be defined by programming the radius (R) or the co-ordinates of the centre (I, J).

e.g.: Clockwise circular interpolation in the XY plane



Circular interpolation can also be programmed in the incremental mode (G91), that is with the co-ordinate of the final point and of the centre of the circumference referred to the starting point (programmed in the previous block of positioning of the axes).

e.g.; Clockwise circular interpolation in the XY plane in incremental mode

*Clockwise circular interpolation in incremental mode with co-ord. I and J of the centre of the circle:*

```
N10 G0 X20 Y20
N15 G91 G2 X30 Y0 I15 J0 F200
```

*Clockwise circular interpolation in incremental mode with the value R for the circle radius:*

```
N10 G0 X20 Y20
N15 G91 G2 X30 Y0 R15 F200
```

If programmed, the radius (R) must have a positive value for an arc greater or equal to 180 degrees and a negative value for an arc of less than 180 degrees.

e.g.; Clockwise circular interpolation in the XY plane as a function of the sign of the radius R

*Programming with positive radius:*

```
N5 G0 X0 Y0
N10 G2 X0 Y-20 R17 F500
( dotted-line circle)
```

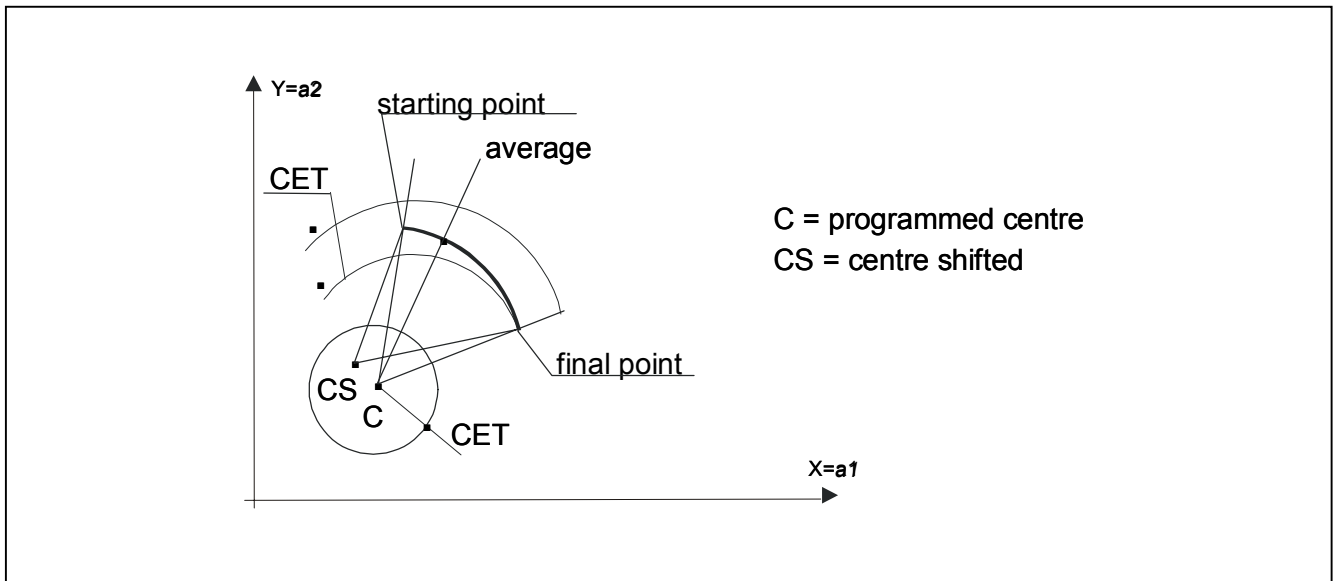
*Programming with negative radius:*

```
N5 G2 X0 Y-20 R-17
( continuous-line circle)
```

The maximum arc that may be programmed is 360 degrees (full circle): in this case the co-ordinates of the centre are always required.



e.g.; Clockwise circular interpolation with approximate co-ordinate of the centre



In circular interpolation the parametre  $@90$  defines the tolerance of the accuracy within which the difference between the initial radius and the final radius of the arc of a circle must remain.

The syntax is the following:

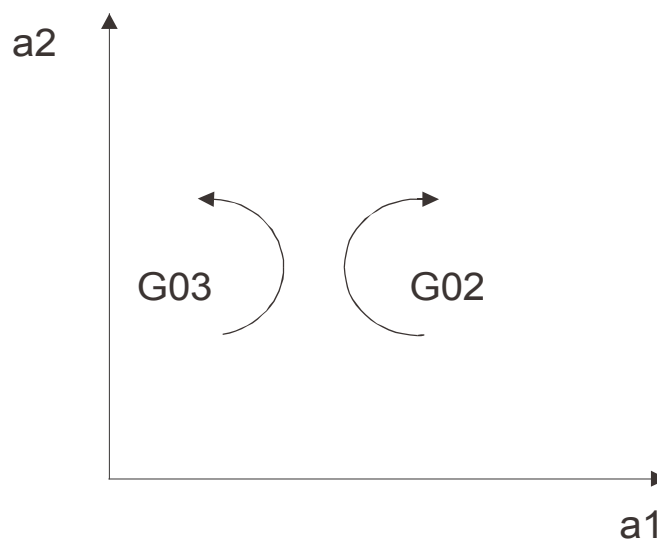
$@90, value$

where *value* is the value of the tolerance of the accuracy expressed in mm.

If the difference between the initial and the final radius is less but not zero, the system effects a geometric average of the data of the circle according to the values specified with the code  $@90$ .

If the difference is greater or equal to the value assigned to  $@90$ , an error is detected and the final programmed points are not executed. If this happens, it is necessary to change the program or increase the tolerance defined by  $@90$ .

The following figure shows the directions of circular interpolation referred to the axes  $a1$  and  $a2$ ; these latter axes must always be at  $90^\circ$  to each other.



Directions for circular interpolation

## G03 (G3) Anticlockwise Circular interpolation

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

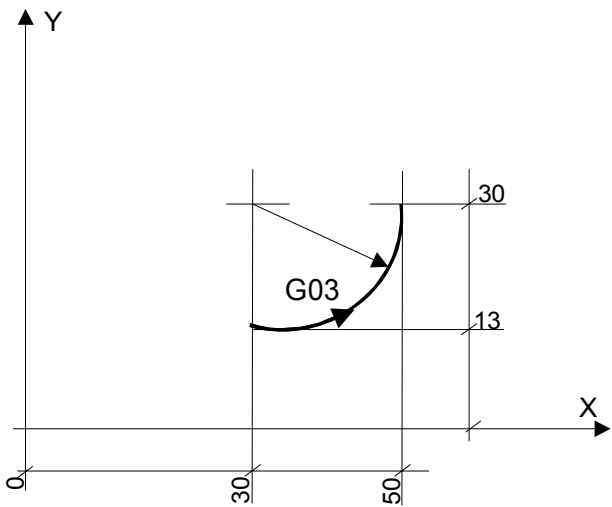
Source	Host computer / Resident program	
Active	Until the execution of instructions G00, G01, G02	
Execution	Queued	
Conditions	Not allowed in the "Positioner" models and in models with only one axis	
Compatible functions	G70, G71, G90, G91	
Syntax	G03 [Compatible G's] [X...] [Y...] [Z...] [W...] [I...,J...] [R...] [F...] X,Y,Z,W: co-ordinate of the final point I: co-ordinate of the centre 1st axis (a1) J: co-ordinate of the centre 2nd axis (a2) R: radius F: velocity of the point	(1) (2) (2) (3)

- Notes:
- (1) The two axes to be programmed are those of the work plane selected (see functions G16,G17,G18,G19).  
The addresses Z and W are allowed only if available on the model of controller selected
  - (2) *a1 and a2 are the axes defined with G16*
  - (3) The radius R is alternative to the co-ordinates I and J: contemporaneous programming of the radius and the central co-ordinate is not allowed.

**Description**

Execute a movement in anticlockwise circular interpolation in the plane previously selected with the functions G16,G17,G18 and G19. The programmed position is arrived at with circular type of movement and co-ordinated on both axes (position reached contemporaneously by both axes). The programming rules are identical to the instruction G02: refer to the drawing and the examples supplied in its description.

e.g.; Anticlockwise Circular interpolation in the XY plane



*Anticlockwise Circular interpolation with the central co-ordinate of the circle I and J:*

```

                    N10 G0 X30 Y13
                    N15 G3 X50 Y30 I30 J30 F500
                
```

*Anticlockwise Circular interpolation with the value for the radius of the circle:*

```

                    N10 G0 X30 Y13
                    N15 G3 X50 Y30 R-20 F200
                
```

## G04 (G4) Hold at end of block

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	<b>(1)</b>
Syntax	G04 F... F: hold time	<b>(2)</b>

- Notes:
- (1) The function G04 does not allow the programming of other ‘G’ functions in the same block.
  - (2) The hold time is expressed in seconds.

**Description**

G04 provokes a hold at the end of the execution of the previous block. The execution of the instruction following a G04 will begin at the end of the hold time programmed at the address F. It can be defined with an accuracy of 0.01sec in the range 0.01 to 9999.99 sec.

e.g.; Execution of a hold time in seconds

```
G01 X123.45 Y125.05 F1000
G04 F3.5 ; 3.5 second hold time between two linear interpolation instructions
G01 X0 Y34.7
```

## G06 (G6) Velocity controlled axis

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	In the block it is programmed	<b>(1)</b>
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G06 [Xs] [Ys] [Zs] [Ws] [F...] Xs,Ys,Zs,Ws: controlled axes F: velocity of the axes	<b>(2)</b>

- Notes:
- (1) During the movement in velocity the axis cannot be instructed with the functions G00, G01, G02 and G03
  - (2) The movement concerns only the axes defined in the instruction  
The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

This instruction, generally used with rotary axes, allows the start or the stop the velocity controlled movement of the axes specified in the instruction.

In the case of a start, the axes continue to move, in the programmed direction and until a request to stop is received, with the velocity programmed (address F) in the block or, if omitted, the last velocity programmed.

The character "s", if present, defines the direction of displacement (start axis)

- '+' : forwards
- '-' : backwards

If the character "s" is omitted, a stop of an axis previously instructed to move with the G06 is understood.

e.g.; Execution of a series of movements in velocity

```
G06 X+ Y- F1000      ;move x-axis forwards and y-axis backward with a maximum velocity of
                    1000 mm/min.
G6 Y Z+ F1500       ;stop the y-axis and move the z-axis forward with a velocity of 1500
                    mm/min.
G06 X Z             ;stop the x- and the z-axes
```

## G16 Define the Plane of Circular interpolation

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G17, G18, G19	
Execution	Queued	
Conditions	Not allowed in the “Positioner” models and in models with only one axis	
Compatible functions	-	
Syntax	G16 [X] [Y] [Z] [W] X,Y,Z,W: axes	<b>(1)</b>

Notes: (1) Only two axes must be defined: the first represents the abscissa and the second the ordinate  
 The Z and W addresses are allowed only if available on the model of controller selected  
 The instruction is not allowed on controllers for only one axis

**Description**

The code G16, and in a similar way to the codes G17, G18 and G19, defines the *a1* and *a2* axes of the plane of interpolation. The following instructions are this equivalent:

- G16 XY = G17
- G16 ZX = G18
- G16 YZ = G19

It is noted that inverting the order of the axes does not invert the direction of displacement of the circular interpolation, clockwise or anticlockwise, (always determined looking at the selected plane from the positive side of the orthogonal axis) but the association of the central co-ordinates is inverted as I is always related to *a1* and J to *a2*.

e.g.:

N...  
 N15 G16 X Z ;specify the plane of interpolation formed by the X- and Z-axes  
 N...

**G17 - G18 - G19 Select the Plane of Circular interpolation**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G16, G17, G18, G19	
Execution	Queued	
Conditions	Not allowed in the "Positioner" models and in models with only one axis	<b>(1)</b>
Compatible functions	-	
Syntax	G17 G18 G19	<b>(2)</b>

- Notes:
- (1) The functions G18 and G19 are allowed only for models with 3 and 4 axes
  - (2) The instructions have no parameters  
The Z and W addresses are allowed only if available on the model of controller selected

**Description**

The functions are used to select one of the three predefined planes of interpolation:

G17 plane XY: I = central co-ordinate X; J = central co-ordinate Y

G18 plane XZ: I = central co-ordinate X; J = central co-ordinate Z

G19 plane YZ: I = central co-ordinate Y; J = central co-ordinate Z

where the first axis represents the abscissa and the second the ordinate.

e.g.; Anticlockwise Circular interpolation in the ZX plane

G00 X0 Z0	;position the axes at the last programmed velocity
G18	;select the XZ plane
G03 X15 Z0 I7,5 J0	;execute a semi-circumference in the anticlockwise direction with a velocity of 1000 mm/min. The centre is at 0 mm along the z-axis (abscissa) and at 7.5 mm along the x-axis (ordinate).

## G20 Unconditioned jump

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Resident program	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Compatible functions	-	
Syntax	G20 L... L: label number	<b>(1)</b>

Notes: (1) The number of the label is programmed at the address L in the range:1 ... 9999

**Description**

The execution of the program that does not continue with the following block but with the instruction to go to the label defined in the instruction. Within a program, a label is defined with the # symbol.

e.g.: Program repeated indefinitely

```

N1 #5                                ;label 5
N2 G01 X50 Y100 F1000
N3 G04 F1
N4 G01 X0 Y0
N5 G20 L5                            ;proceed to execute block No. 3
N6 M02
    
```

## G21 - G22 Conditioned jump

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Resident program	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Compatible functions	-	
Syntax	G21 L... G22 L... L: label number	<b>(1)</b>

Notes: (1) The number of the label is programmed at the address L in the range:1 ... 9999

**Description**

The jump to the programmed label, is executed or not as a function of the state TRUE or FALSE of the flag programmed that in its turn is determined in a previous programmed block with the code G69 or by a mathematical function.

The code G21 executes the jump only if the flag is TRUE while the code G22 executes it only if the flag is FALSE.

If the jump is not executed, the program continues to the block of program immediately following the code G21 or G22.

Within a program, a label is defined with the # symbol.

e.g.; Program repeated until an input is activated

```

N1 #40                                ;label 40
N2 G01 X50 Y100 F1000
N3 G04 F1
N4 G01 X0 Y0
N5 G69 P3                             ;assign the state of input 3 to the flag
N6 G21 L50  if flag = true           ;if true (input active) jump to block No.8, otherwise execute No.7
N7 G20 L40                             ;proceed to execute block No.3
N8 #50                                ;label 50
N9 M02  ←-----                    ;end of program
    
```



## G25 - G26 Define Field of Work limits

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

### Properties

Source	Host Computer / Resident program	
Active	Until the execution of instructions G27	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G25 [X(lim.)] [Y(lim.)] [Z(lim.)] [W(lim.)] X(lim.), Y(lim.), Z(lim.), W(lim.) : limit co-ordinates	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 X Y Z and W are always absolute values referred to the origin of the active axis,

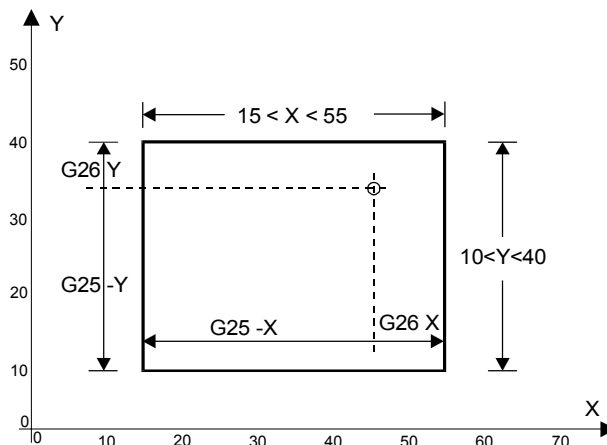
### Description

The functions G25 (minimum limits) and G26 (maximum limits) are used to define the co-ordinates of the limits of the field of work of the axes in order to avoid that, during the work, the axes find themselves in an area where there is danger of a collision.

The limits defined with the machine parameters (instructions @60 and @61) can only be reduced with the functions G25 and G26 and not increased.

The co-ordinates programmed as limits are always referred to the origin of the “active” axis.

e.g.; Define the limits of the field of work



```

N5 G0 X45 Y35 Z20 ;position on the zero point of the workpiece
N6 G53 XY

N7 G25 X-30 Y-25 Z-15 ;lower limit
N8 G26 X10 Y5 Z10 ; UPPER LIMIT
    
```

## G27 Cancel Field of Work limits

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G25 and G26	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G27 [X] [Y] [Z] [W] X,Y,Z,W: axes names	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 The cancellation of the limits is only made for the axes defined in the instruction

**Description**

The instruction cancels the limits programmed with the functions G25 and G26, re-activating the limits of work defined with the machine parameters (instructions @60 and @61).

e.g.; cancel the limits of the field of work

```
N...
N10 G0 X400 Y500 Z600 ;position on the zero point of the workpiece
N15 G53 XY ;displacement of the origin
N20 G25 X-300 Y-300 Z-400 ;Define lower limits
N25 G26 X200 Y250 Z200 ;Define upper limits
N...
```

PROGRAM WORKPIECES

```
N...
G27 XYZ ;annul limits
N...
```

## G30 Recall a Subprogram

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Resident program	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Compatible functions	-	
Syntax for calling an internal subprogram	G30 L...[, (r)] L: subprogram label r: repetition factor	
Syntax for calling an external program	G30 H...[, (r)] H: program number r: repetition factor	

**Description**

The G30 L code enables the running of parts of a program between a label (for example :5) and the code M99. These parts are subprograms and must be inserted after the M02 code of the main program. The labels are numbered between 1 and 9999, preceded by the sign :.

The code G30 H allows the execution of a program that is different to that being currently executed as if it were a subprogram of the former.

It is possible to call up another subprogram from inside a subprogram up to 8 levels of nesting. The block executed after the G30 code is the first block of the subprogram called.

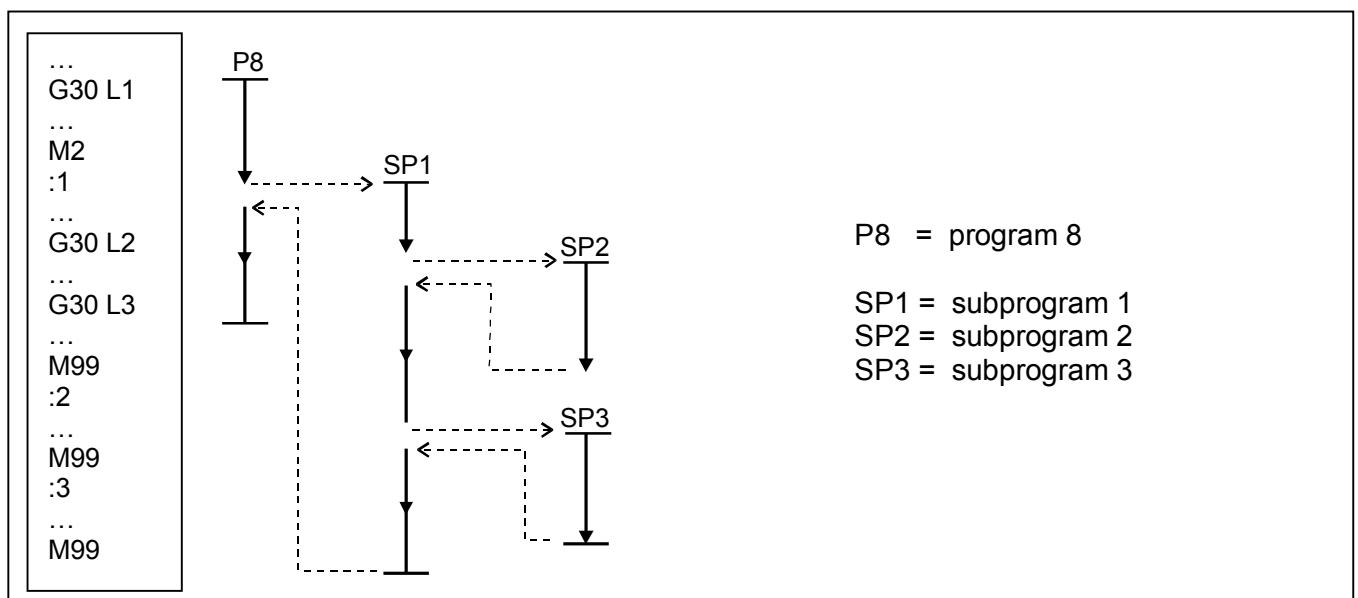
The execution of the subprogram can be repeated more than once, defining in the code the number of repetitions (r). r can assume any value between 1 and 99.

On returning from the subprogram (after (r) + 1 executions of the same), the execution proceeds with the block following the G30 code.

e.g.: Codes for calling subprograms

- G30 L5 ;subprogram 5 is executed only once
- G30 L27,9 ;recall subprogram 27 and execute it 10 times (9 repetitions)
- G30 H85 ;recall program 85 and execute it only once

e.g.: nesting of 3 subprograms



## G31 - G32 Conditioned recall of Subprogram

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Resident program	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Compatible functions	-	
Syntax for calling an internal subprogram	G31 L...[, (r)] G32 L...[, (r)] L: number of subprogram r: repetition factor	<b>(1)</b>
Syntax for calling an external program	G31 H...[, (r)] G32 H...[, (r)] H: number of program r: repetition factor	

**Description**

This function operates exactly as described for the G30 instruction, but the controller is passed to the subprogram indicated according to the state, TRUE or FALSE, of the program flag that in turn is determined by executing a block of program previous to the G69 code or a mathematical function. Code 31 recalls the subprogram if the flag is TRUE while code G32 calls it if the flag is FALSE. If the subprogram is not recalled, the program block following the G31 or G32 code will be executed. The execution of the subprogram can be repeated more than once, defining the number after the code for the repetition factor (r). On returning from the subprogram (after (r) + 1 executions of the subprogram), the execution continues with the block that follows the code G31 or G32.

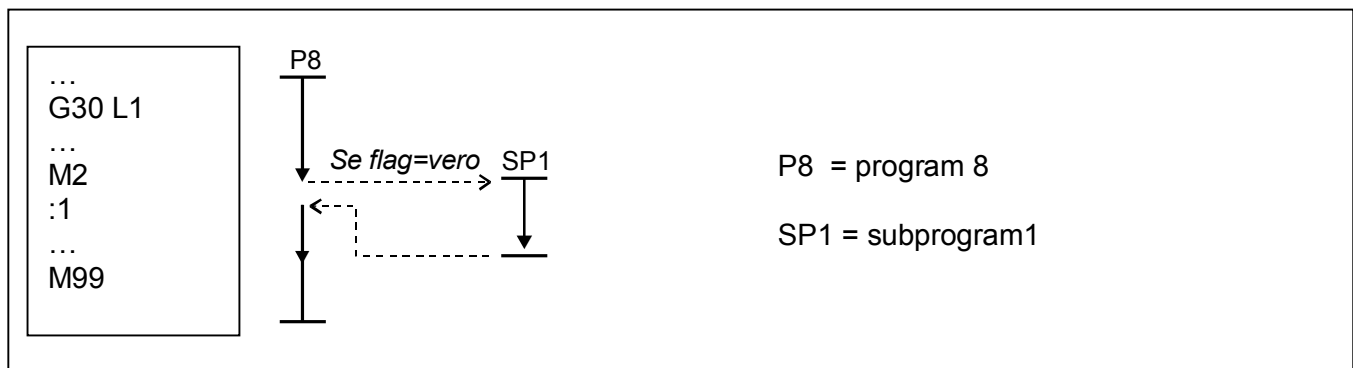
e.g.: Subprogram 5 is recalled only if input 8 is active

```
G69 P8           ;assign the state of input 8 to the flag
G31 L5          ;if true (input 8 active) recall subprogram 5 and execute it only once.
```

e.g.: Program 85 is recalled and executed 5 times only if input 8 is not active

```
G69 P8           ;assign the state of input 8 to the flag
G32 H85,4       ;if false (input 8 not active) recall program 85 and ;execute it 5 times
                 (i.e. with 4 repetitions)
```

e.g.: nesting of 1 subprogram



## G50 Cancel Displacement of Origin of the Axes

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G52 and G53	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G50 [X] [Y] [Z] [W] X, Y, Z, W: names of axes	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 The cancellation of the displacement of the origins is made only for the axes defined in the instruction

**Description**

The function G50 is used to annul the displacement of the absolute origins defined with the functions G52 and G53 only for the axes defined in the instruction,.  
 The origin of the absolute co-ordinate returns to the point of machine zero determined by the function G51 (Set Point) or the function G54.

e.g.; cancel displacement of origins of the axes

```
N10 G52 X10 Y20 ;displacement of origins of X- and Y-axes
N11 G22 L3 ;recall of the workpiece program
N...
N...
N15 G50 X ;cancel displacement of X-axis origin
```

## G51 Seek origin of axes (set-point)

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G52 and G53	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G51 [X(s)] [Y(s)] [Z(s)] [W(s)] [F...] X(s), Y(s), Z(s), W(s): axes and direction F: target velocity	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

With the function G51, the axes defined in the instruction move in the programmed direction, forward (+) or backward (-), seeking the machine zero point.

E machine zero point is reached independently by each axis at the target velocity programmed in the block at address F or, if omitted, at the last programmed velocity.

The character hat follows determines the direction of displacement of the axes:

‘+’ = axis forwards

‘-’ = axis backwards

Optionally, the target velocity can be programmed at the address F in the same block.

F is always an absolute value expressed in mm/min (G71) or in inches/min (G70).

After the execution of the set-point, the machine origin can be shifted with the function G52.

The software limits programmed with @60 and @61 are always referred to the machine zero.

e.g.; seek the axis 0 in directions X+, Y-

```
N...
N5   G51  X+   Y-
N...
```

e.g.; seek the axis 0 in direction Z+

```
N...
N5   G51  Z+ F2000
N...
```

e.g.; seek the axis 0 in directions X+, Y+ and Z-

```
N...
N5   G51  X+   Y+   Z-
N...
```

## G52 Displacement of Axes Origin

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

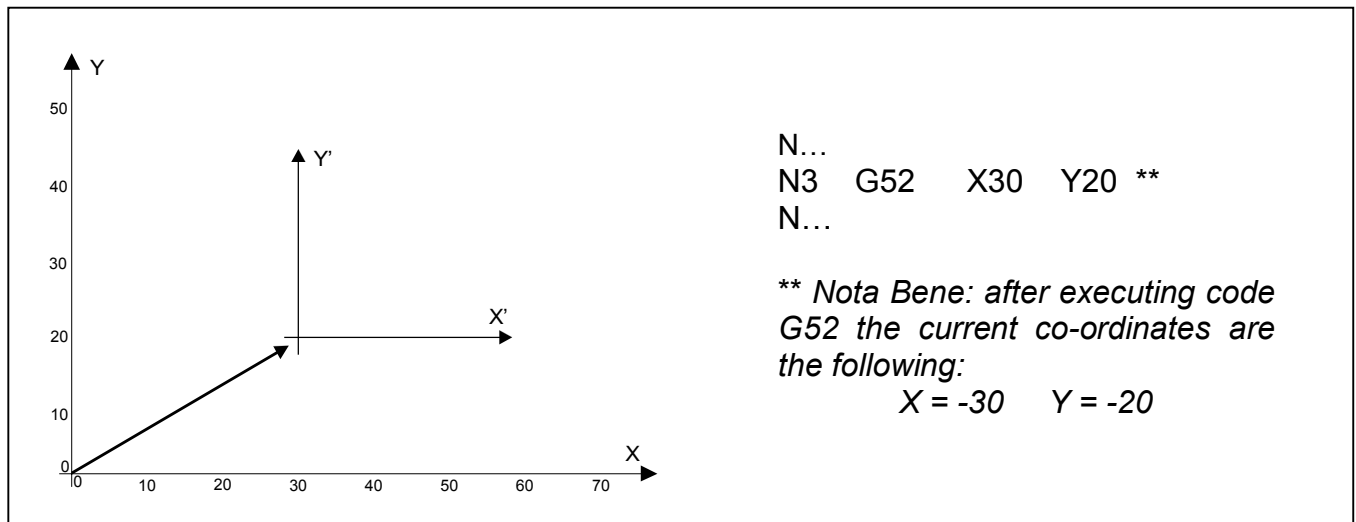
Source	Host Computer / Resident program	
Active	Until the execution of instructions G50, G51 and G53	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G52 [X(ori)] [Y(ori)] [Z(ori)] [W(ori)] X(ori), Y(ori), Z(ori), W(ori): new origin	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 The programmed co-ordinates are always referred to the active origin in that moment.

**Description**

Using the facility to displace the origin, means that the same program can be used at different points on the workpiece without changing it. The function G52 redefines the origins of e absolute co-ordinates only for the axes defined in the instruction, shifting them with respect to the actual origin.  
 The function G50 annuls the all displacement of the origin defined by G52 and G53

e.g.; displacement of the origin of the axes



## G53 Origins of Axes at this Point

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

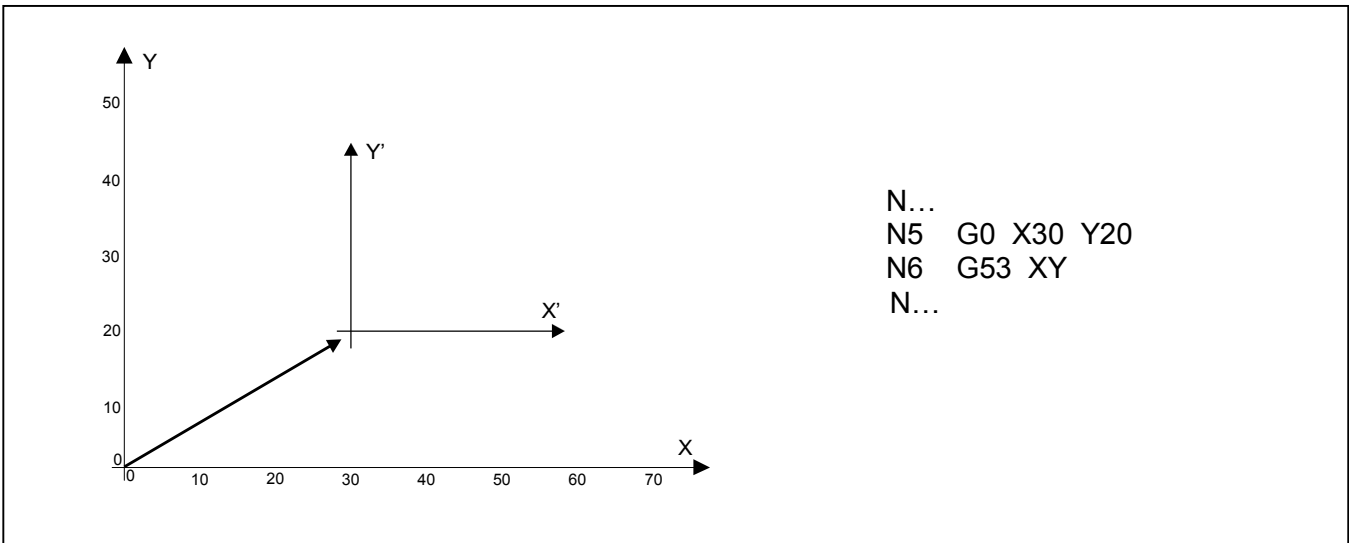
**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G50, G51 and G52	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G53 [X] [Y] [Z] [W] X, Y, Z, W :names of axes	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 The displacement of the origins is only made for the axes defined in the instruction

**Description**

The function G53 can be used to shift the origin of the axes .  
 The co-ordinate at which the axes defined in the instruction are currently positioned is considered to be the new origin.  
 This displacement can be annulled with the instruction G50.





<b>G54 Origins of Axes at this Point (software set-point)</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G51 and G52	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G54 [X(org)] [Y(org)] [Z(org)] [W(org)] "org": new position	<b>(1)</b>

Notes: (1) At least one of the addresses X Y Z or W must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected  
 The displacement of the origins is made only for the axes defined in the instruction

**Description**

The function G54 can be used to shift the origin of the axes .  
 The co-ordinate at which the axes defined in the instruction are currently positioned is considered to be the new origin.  
 G54 cannot be annulled with instruction G50.  
 Function G54 is equivalent to the function G51, but it is executed in software, without any hardware support.  
 The execution of the code G54 annuls the effects of the code G51, if it has been executed previously.  
 With the function G54 the axis or the all axes specified will be found to the same co-ordinate.

Example:

G54 X100      The actual position of the axis X now is 100

If the specific axis isn't programmed with a co-ordinate, this is 0 for default (G54 X is the same of G54 X0)

## G55 – G56 Save – Restore Origin

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

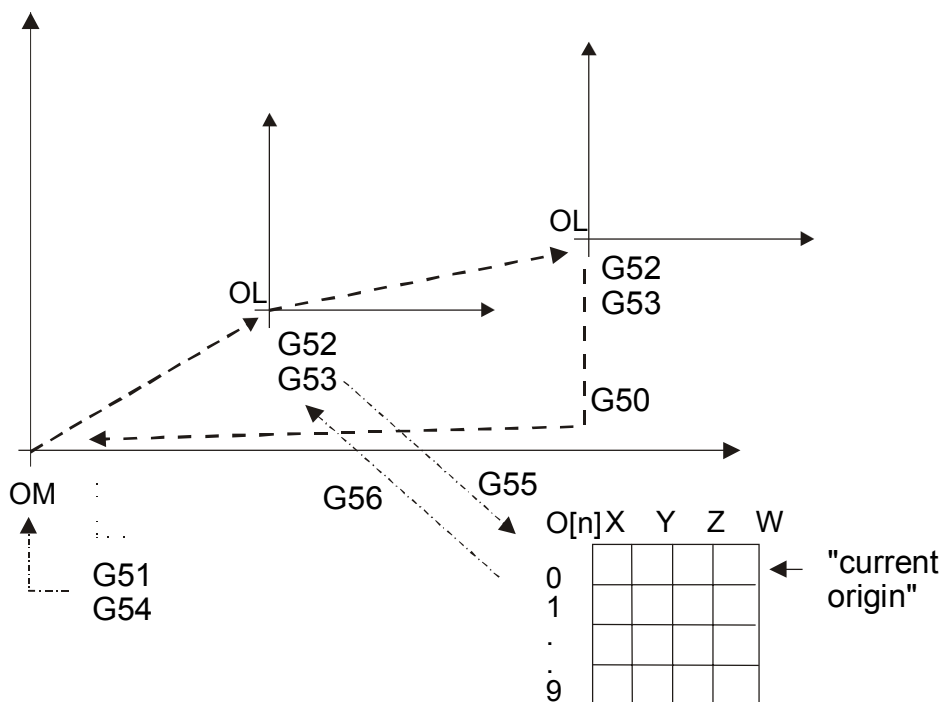
**Properties**

Source	Host Computer / Resident program	
Active	G55: until re-written by G55 G56: Until the execution of instructions G51 G52 G56	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G55 O[numorg] G56 O[numorg] "numorg": position of the origin [0...9]	(1) (1)

Notes: (1) If no parameter is specified, O[0] is understood.  
The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

The instruction G55 is used to save the current origin in the table O[ ].  
The values of the origins sad with this instruction G55 and be recalled, with respect to the machine zero (OM), using instruction G56.  
O[0] always contains the last local origin set with G52 and G53.  
If an origin is saved in a position that is already used, the old origin will be overwritten and the relative data lost.



To create the preset of the origins table use the following instructions:

O[numorg] = <axis> <val> [ ,<axis> <val>] ... [ ,<axis> <val>]

For example: O[5] = X20, Z30, W43

e.g.; series of instructions

```
G51 X+ Y+ Z+ W+      ;set OM
G52 X100 Z400        ;OL (local origin)= (100,0,400,0) ; = O[0]
G55 O[4]              ;save OL in O[4]
...
G51 X+ Y+ Z+ W+      ;set OM
G56 O[4]              ;OL = (100, 0, 400, 0)
```

## G57 – G58 Activate - Disactivate Automatic Linkage

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G58/G57 and G61	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G57 [, (n)] G58	

**Description**

The instruction G57 activates and the instruction G58 deactivates the automatic linkage between two segments of consecutive trajectories executed in linear or circular interpolation; the linkage is therefore only valid for movements G1, G2 and G3.

The optional parameter can have any value between 1 and 10 and it allows the choice of with what precision the controller maintains the programmed trajectory according to the following table.

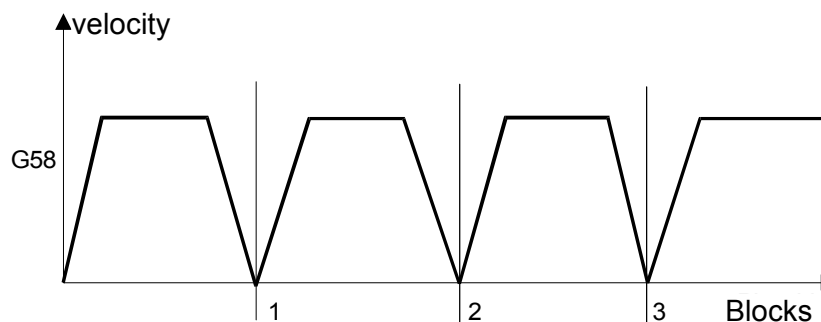
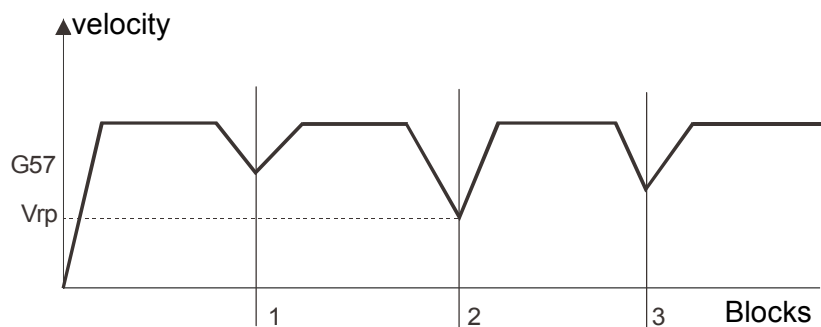
1 = The linkage is executed only if it does not alter the trajectory

...  
...

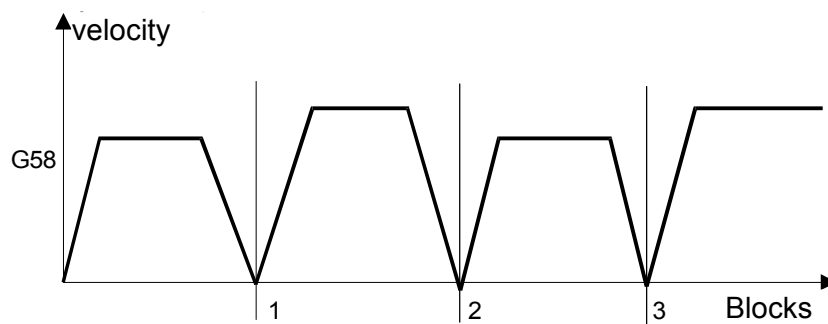
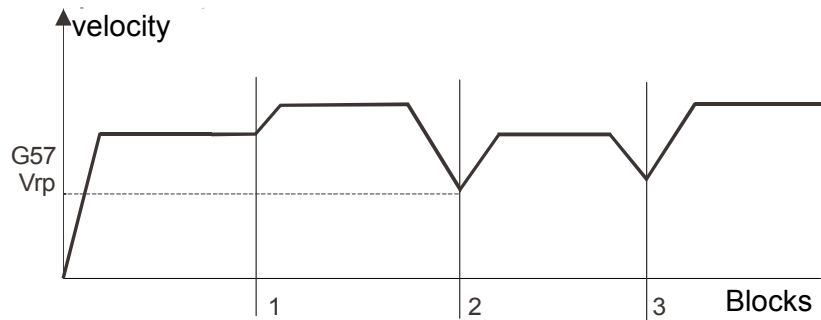
10 = The linkage is executed anyway, giving priority to maintaining the speed rather than respecting the programmed trajectory.

If the parameter is omitted, a G57 instruction will be executed with the automatic linkage with the last value assigned to the parameter. The default value on power up is 1.

e.g.; diagram showing how the codes G57 and G58 work when the programmed velocity is the same for the whole profile



e.g.; diagram showing how the codes G57 and G58 work when the programmed velocity is changed for every block of the profile



$V_{rp}$  depends on different factors such as the angle between the two displacements, their length and acceleration, and the value assigned to the optional parameter of the G57 instruction.

e.g.; use of the G57 and G58

G57

G1 X.. Y.. F..

G3 X.. Y.. R..

G1 X..

G58

G2 X.. Y.. R..

## G61–G62 Activate-Disactivate “Accurate Stop (in position)”

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

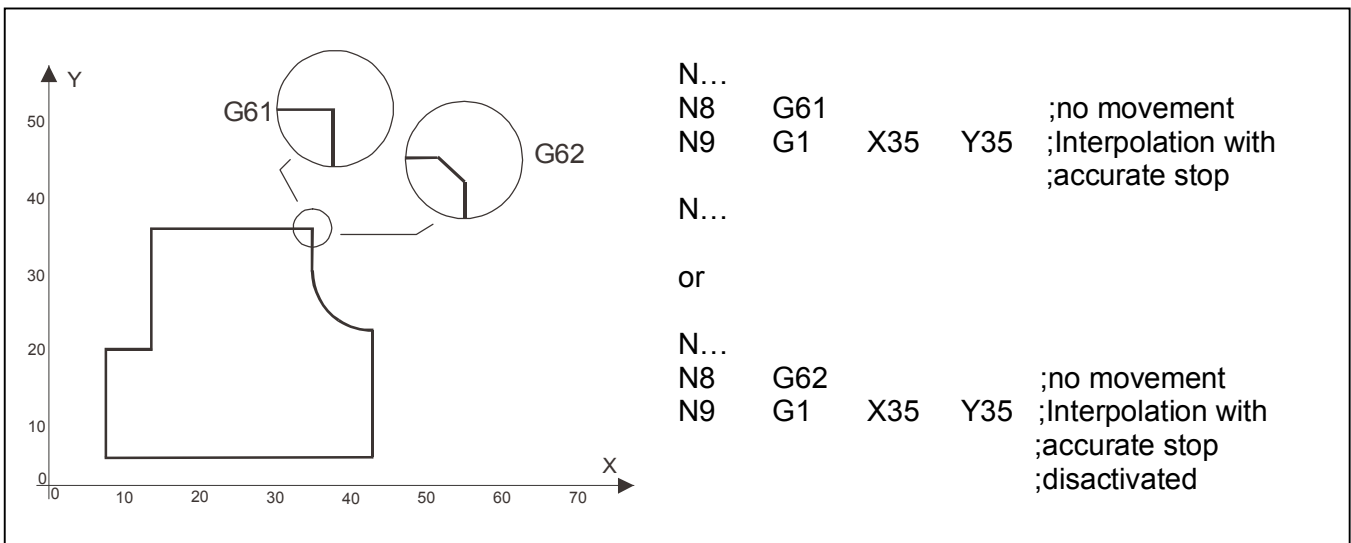
**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G62/G61 and G57	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G61 G62	

**Description**

The function G61 is used to make the controller wait until the “IN POSITION” window programmed with a machine parameter (instruction @59) is reached before continuing to execute the next block. Function G62 deactivates the accurate stop (see function G61).

e.g.: Execution of the two instructions



## G63 – G64 Not Wait (Wait) for End of Movement

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until the execution of instructions G64/G63	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G63 G64	

**Description**

Function G63 instructs the controller not to wait for the previous instructions to be completed before executing instructions that not imply axes movement.

Function G64 imposes that instructions being executed are terminated before any other instruction is executed.

e.g.; programming the two cases

```

N...
N5   G63
N6   G0 X.. Y..
N7   G67 P0           ;function G67 is executed before instruction G0 of the block
                        ;N6 is terminated.

N8   G64
N9   G1 X.. Y..
N10  G68 P0           ;function G68 is executed only after the code G1 of the
                        ;block N9 is terminated
    
```

## G65-G66 Hold Input / Status Bit

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G65 [P(c)] [XYZW(i)] [D(b)] [O(n)] [E(a)]	<b>(1)</b>

Notes (1) At least one of the addresses P, X, Y, Z, W, D, E, O  
 The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

The functions G65 and G66 enabled the controller to wait until the logic condition 0 (G65) or 1 (G66) is verified at the input states and bit status register.

The code remains in execution until all the conditions defined in the block are verified.

P(c) : PLC INPUT / CONTROLLER INPUTS (for models without PLC)  
 (value in the range 0..31)

X(i), Y(i), Z(i), W(i): axes INPUTS  
 the value in the range 0..31 defines the input:  
 0 = not used  
 1 = not used  
 2 = not used  
 3 = emergency  
 4 = limit switch –  
 5 = limit switch +  
 6 = motor drive o.k.  
 7 = machine zero  
 - refer to instruction ^O for the description of the inputs

D(b): BIT OF STATUS REGISTER “D” (CNC) (for the meaning of the individual bits, see the Ctrl A instruction)  
 The value in the range 0..31 defines the bit of the status register “D” (CNC)

DX(b), DY(B), DZ(b), DW(b): BIT OF AXIS STATUS REGISTER AXIS “X”, “Y”, “Z”, “W”(for the meaning of the individual bits, see the Ctrl A instruction)  
 The value in the range 0..31 defines the bit of the status register for X-, Y-, Z-, W-AXIS

E(a): BIT OF ALARM REGISTER “E” (CNC) (for the meaning of the individual bits, see the Ctrl D instruction)  
 The value in the range 0..31 defines the bit of the alarm register “E” (CNC)

EX(a), EY(a), EZ(a), EW(a): BIT OF AXIS ALARM REGISTER “X”, “Y”, “Z”, “W” (for the meaning of the individual bits, see the Ctrl D instruction)  
 The value in the range 0..31 defines the bit of the alarm register for X-, Y-, Z-, W-AXIS

O(n): COMMUNICATION CHANNEL  
 (value in the range 1..9)  
 1 = ^Z  
 2..9 : not active



e.g.:

G66 P4 Y5            wait for input 4 of the PLC and limit switch forwards of the Y-axis in state 1

e.g.:

G65 X6 Y6 Z6        wait for inputs of motor drive O.K. of all the axes in the state 0

e.g.:

G65 O1                wait for <SUB> code

## G67-G68 Handling Devices

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	In the block it is programmed	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G67 [P(r)] [XYZW(o)] [O(n)] [H(n)]	<b>(1)</b>

Notes (1) At least one of the addresses P, X, Y, Z, W and O must be defined  
 The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

The functions G67 and G68 allow the activation of controller outputs by placing them in the logic state 1 (G68) or 0 (G67) and to activate (G68) or disactivate (G67) some internal functions of the controller. The interested device is specified by the address (P, X, Y, Z, W, O, H).

P(r): HANDLING OF POINTS (D) PLC OR THE ACTUAL OUTPUTS (if the control does not include a PLC)

r = 0 to 31

example: **G68 P3** activates output 4 of the controller (in the documentation of the hardware, the outputs are numbered from 1 and onwards) or sets point 3(D) of the PLC.

X(o), Y(o), Z(o), W(o): HANDLING THE OUTPUTS OF THE AXES

o = 1 : drive unit direction: instruction executed only if the handling of the DIR signal has not been activated by the controller.

o = 2 : enable drive unit: instruction executed only if the handling of the ENABLE or DISABLE signal is not activated by the controller.

Example **G67 X1** disactivates the DIR output of the X-axis.

O(n) (...): HANDLING THE COMMUNICATION CHANNEL (only for the instruction G68)

**G68 O1** : transmits the character <^Z> (1Ah) to the serial communications channel. This enables the external device to synchronise with the execution of the program.

**G68 O2**, “(message)”, (s). Sends the specified message over the serial communications channel and assigns the control state.

s = 0 : control state unchanged

s = 1 : stop CNC

s = 2 : emergency

s = 3 : stop axes

H(n): HANDLING OPTIONAL DEVICES

**G68 H4**, (polarity), (slow distance), (impact velocity): enables the probe with specific parameters. The polarity can only assume the values of 0 or 1: 0 means that the transition from 1 to 0 of the input to the probe will identify the valid point.

The slow distance is expressed in engineering units and specifies a terminal tract of the distance to be covered at reduced speed to increase the accuracy when identifying the co-ordinate. The speed at which the terminal tract is executed is the impact velocity which is expressed in the same units as the other velocities in the CNC program. The probe function is not enabled. The probe option is not available on Goya.

**G67 H4** : disables the probe.

**G68 H5**, (c): enables the probe. In the course of the first movement executed after this instruction, the co-ordinates at which the axes find themselves at the instant the probe is presented with a valid front will be saved in the variables QAX, QAY,....

c = 0 : the movement will be interrupted at the co-ordinate probed.

C = 1 : the movement will proceed until the objective is reached. If the controller detects the co-ordinate before arriving at the beginning of the slow distance, an error signal will be generated and the controller will go into emergency.

If the probe does not detect a valid signal, the bit concerned in the status register will not be activated.

To acquire a new position of the probe it will be necessary to re-arm the probe with the **G68 H5** instruction.

**G67 H5** : disables the probe.

**G67 H6**, (m): defines the mode of application.

m = 0 : do not execute either the initial tract nor the final tract.

m = 1 : execute the initial tract but not the final tract.

m = 2 : execute the final tract but not the initial tract.

m = 3 : execute both the initial tract and the final tract.

**G68 H6**, (d), (s), (t), (r): enable and disable the mode of application.

d = 0 : mode disabled.

d = 1 : continuous mode.

d = 2 : reserved for future improvements.

d = 3 : mode enabled at points.

s is the distance between the points in engineering units, if the mode is enabled at points, or the length between the extremes, if the continuous mode is chosen.

t is the time of activation of the output required to form a point. It is expressed in milliseconds between 0 and 10000.

r is the delay with which the output is activated with respect to the theoretical instant in which it should be enabled. It is expressed in milliseconds between 0 and 10000.

**G68 H7**, (n), (period), (resolution): enables and disables the PWM function on a logic output, defining the period and the resolution of the signal that will be generated. Regarding the regulation of the duty cycle, the output will then be handled as an analogue output, using the instructions for control of the spindle:

**@52** for the CNC or handling analogue outputs for the PLC

n specifies the number of the output. In the Goya, that has a dedicated connector for the PWM output, n can only have the value 3. For the Picasso 2000 and Rubens, n can have any value between 0 and 3, which correspond to output 1 to 4. In any case no more than one PWM output can be handled at any one time.

Period and resolution are two parameters whose values must be obtained from the tables of the PWM handling included as an appendix of the documentation of the chosen controller. If all the three parameters are 0, the PWM function is disabled.

**N.B.:** The instructions for handling the optional functions are operational only if the controller includes the options concerned.

## G69 Assign State to the “Flag”

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	In the block it is programmed	
Execution	From program: Queued	
Conditions	-	
Compatible functions	-	
Syntax	G69 [P(c)] [XYZW(i)] [D(b)] [O(n)] [E(a)] [(u)]	<b>(1)</b>

Notes (1) At least one of the addresses P, X, Y, Z, W, D, E, \ and O must be defined. The addresses Y, Z and W are allowed only if available on the model of controller selected

**Description**

Function G69 is used to assign the state 0 or 1 to the program “FLAG” used by the conditional jump (G21, G22) and call subprogram (G31, G32) functions. The state assigned to the flag is that of the input (addresses P, X, Y, Z, W) or of the bit of the status register (address D) defined in the instruction, or it will be related to the condition “received” (logic state 1) o “not received” (logic state 0) for what regards the synchronisation code Ctrl-Z (address O: communication channel).

P(c) INPUT PLC / INPUTS CONTROLLER (if model without PLC)  
value in the range 0 .. 31

X(i), Y(i), Z(i), W(i): axes INPUTS  
the value in the range 0..31 defines the input:  
0 = not used  
1 = not used  
2 = not used  
3 = emergency  
4 = limit switch –  
5 = limit switch +  
6 = motor drive o.k.  
7 = machine zero  
- refer to instruction ^O for the description of the inputs

D(b): STATUS REGISTER BIT “D” (for the meaning of the individual bits, see the Ctrl A instruction)

The value in the range 0..31 defines the bit in the status register “D” with the following meaning:  
0 - 7 = bit 0 - 7 of the status register of the Z-axis  
8 - 15 = bit 0 - 7 of the status register of the Y-axis  
16 - 23 = bit 0 - 7 of the status register of the X-axis  
24 - 31 = bit 0 - 7 of the status register of the CNC

DX(b), DY(B), DZ(b), DW(b): BIT of STATUS REGISTER for “X”, “Y”, “Z”, “W” AXIS (for the meaning of the individual bits, see the Ctrl A instruction)

The value in the range 0..31 defines the bit of the status register X,Y,Z,W-AXIS

E(a): ALARM REGISTER BIT “E” (for the meaning of the individual bits, see the Ctrl D instruction)

The value in the range 0..31 defines the bit in the alarm register “E” with the following meaning:  
0 - 7 = bit 0 - 7 of the alarm register of the Z-axis  
8 - 15 = bit 0 - 7 of the alarm register of the Y-axis  
16 - 23 = bit 0 - 7 of the alarm register of the X-axis  
24 - 31 = bit 0 - 7 of the alarm register of the CNC

E(a): BIT of ALARM REGISTER “E” (CNC) (for the meaning of the individual bits, see the Ctrl D instruction)  
The value in the range 0..31 defines the bit of the alarm register “E” (CNC)

EX(a), EY(a), EZ(a), EW(a): BIT of ALARM REGISTER for “X”, “Y”, “Z”, “W” AXIS (for the meaning of the individual bits, see the Ctrl D instruction)  
The value in the range 0..31 defines the bit of the alarm register for X,Y,Z,W-AXIS

O(n) COMMUNICATION CHANNEL  
value in the range 1 .. 9  
1 = ^Z  
2 ... 9 : not active

\(u)  
the value in the range 0 .. 7 defines the bit of the condition register:  
0 .. 7 = bit 0 .. 7 of the condition register

e.g.:

G69 P4 ;assign the state of input 4 of the PLC to the flag

e.g.:

G69 Y6 ;assign the state of the input of motor drive O.K. of the Y-axis to the flag

e.g.:

1 = RECEIVED  
0 = NOT RECEIVED

G69 O1 ;assign the state of the <SUB> code to the flag

## G70 G71 Unit of Measurement

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G70, G71	<b>(1)</b>
Execution	Queued	
Conditions	-	
Compatible functions	G00, G01, G02, G03, G90, G91	
Syntax	G70 [Compatible G's] G71 [Compatible G's]	

Notes: (1) G70 and G71 annul each other.  
On power-up the code G71 is activated automatically

**Description**

Defines the unit of measurement of the controller:

G70 sets the unit of measurement to inches

G71 sets the unit of measurement to millimetres

Switching from one unit to the other, all the information regarding position and velocity of movement are converted into the appropriate unit of measurement.

e.g.:

N...

N10 G70

N11 G00 X100 G71 Y-50 ;positioning and programming of X in inches and Y in millimetres

N...

## G74 Enabling and disabling the co-ordinate reading by laser

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	In the programmed block	
Execution	Queued	
Conditions	-	
Compatible Functions	-	
Syntax	G74 [XYZW] (n)	<b>(1)</b>

Notes (1) Only one of the addresses X, Y, Z or W must be defined.  
 The addresses Y, Z and W are allowed only if available on the model of controller chosen

**Description**

The data supplied by the LASER measuring device are received on a dedicated serial link. From the moment the reading is enabled, it will be accessible in the variable in which the value of the actual co-ordinate of the specified axis is usually kept. The data supplied by the measuring device are not processed by the controller in any way, as are the co-ordinates of the axes: the values read are therefore to be considered to be expressed in "LASER points".

The parameter (n) selects the devices that can be connected, that, at the time of printing this document, are listed in the following table.

<b>Parameter Value</b>	<b>Effect</b>
0	Disables the LASER reading. The variable reads the real co-ordinates of the axes indicated.
1	Enables the LASER reading with the protocol of the MEL device
2	Enables the LASER reading with the protocol of the ODS2xx device
3	Enables the LASER reading with the protocol of the ODS1xx device

**N.B.:** The instruction interacts with the special function &52 and with the immediate instruction ^A (see the descriptions in this manual).

&52: the activation of the LASER reading of the co-ordinates determines that the data packet is composed according to mode 1.

^A: the activation of the LASER reading of the co-ordinates determines that the four least significant bits of the status byte of the axis, on which the LASER co-ordinates are readable, contain the status of the measurement device.

Example:

```

N...
N10 G74 Z2           ; the data of a LASER type ODS2xx will be available in the variable QRZ
N11 Q136=QRZ        ; reading of the current co-ordinate measured by the LASER
N12 G91 G0 X1       ; displacement of a millimetre on the X-axis
N13 Q136 = QRZ-Q136 ; calculation of the variation of the co-ordinate measured by the effect of the
                    ; displacement
N14 G74 Z0          ; in QRZ the real position of the Z-axis is once more available
N...
    
```

**G80 Enable and Disable Cam Table and Engage Posn.**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the instruction is disabled.	
Execution	Queued	
Conditions	Instruction @20 must have been defined and all the cam points must have been defined	
Compatible functions	-	
Syntax	G80 (E<n.tab> ( , <EN0>..[ , ENn]))	<b>(1)</b>

Notes: (1) n.tab = number of table to enable  
 EN0..n = Engage position for every slave set

**Description**

Enable and disable the tables for the electric cam function.  
 Also indicates the engage position for each individual slave axis.  
 The instruction recalculates the data regarding linkages only if they have been modified by the data.  
 If only G80 is given without any parameter, the table currently in use is disabled.

Example:

G80E1,0,0 ;enable electric axes on table 1 and engage positions of Engage = 0  
 G80E0,0,0 ;enable electric axes on table 0 and engage position = 0

**Notes**

The instruction only works if the controller is fitted with the relative option.



## G81 Define DISENGAGE Positions of Cam

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until disabled with G80	
Execution	Queued	
Conditions	The cam must be active	
Compatible functions	-	
Syntax	G81E<EQ0>..[,Eqn]	<b>(1)</b>

Notes: (1) EQ0..n = disengage position for every slave set [q]

**Description**

For every slave, the instruction defines the position, referred to the master axis, in which the slave should be released from the cam function.

The instruction requires the definition of the disengage positions for all the slave axes defined.

Example:

<pre>@20E0,2,0,XYZ E0[1]=1,2,1 G80E0,0,0 G81E0.5,2</pre>	<pre>;Define the parameters of the cam table ;construct the table ;enable the table ;stop the Y and Z-axes when the master axis (X in this example) ;has reached the positions of 0,5 and 2 respectively</pre>
--	--

**Notes**

The instruction only works if the controller is fitted with the relative option

## G82 Define Automatic Cam Table

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	-	
Execution	Queued	
Conditions	The instruction @20 must have been defined	
Compatible functions	-	
Syntax	G82 E<n.tab>, <pM> , <dM>	<b>(1)</b>

Notes: (1) n.tab = number of table to set  
 pM = Master position first point in table  
 dM = delta Master position

**Description**

The instruction automatically defines the positions of the Master axis in the table, starting from the position pM and with increments of dM. All the records configured in the table are filled in.

Example:

```
@20E0,0,100      ;Define the parameters of the cam table
G82E0,0,100      ;Define all the positions of the master axis (the first is in '0' and
                  ;those that follow are obtained with increments of 100)
```

**Notes**

The instruction only works if the controller is fitted with the relative option

<b>G83 Define Cam K Factor</b>
--------------------------------

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	-	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G83 E<K>	<b>(1)</b>

Notes: (1) K = K factor (the value must be different from 0)

**Description**

The instruction defines a proportionality factor (K) between the velocity of the master axis and the velocity of the slave axes.

Example: Define a proportionality factor of '2'

G83E2

**Notes**

The instruction only works if the controller is fitted with the relative option

## G84 Define Cam Velocity Variations in %

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	-	
Execution	Queued	
Conditions	The instruction @20 must have been defined and the table must have been constructed	
Compatible functions	-	
Syntax	G84 E<V0> , <Vx>	<b>(1)</b>

Notes: (1) Vx = % velocity for each slave axis

**Description**

This instruction can be used to modify the velocity defined in the table of all the slave axes in percentage, without having to modify all the records.

Example:

G84E10,-30 ;increase all the velocities of the first slave axis by 10% and decrease all the velocities of the second slave axis by 30%

**Notes**

The instruction only works if the controller is fitted with the relative option

## G90 G91 Programming Absolute or Incremental Co-ordinates

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G90, G91	<b>(1)</b>
Execution	Queued	
Conditions	-	
Compatible functions	G00, G01, G02, G03, G70, G71	
Syntax	G90 [Compatible G's] G91 [Compatible G's]	

Notes: (1) G90 and G91 annul each other.  
On power-up the code G90 is activated automatically

**Description**

They define the mode of programming the co-ordinates:

G90 active absolute programming

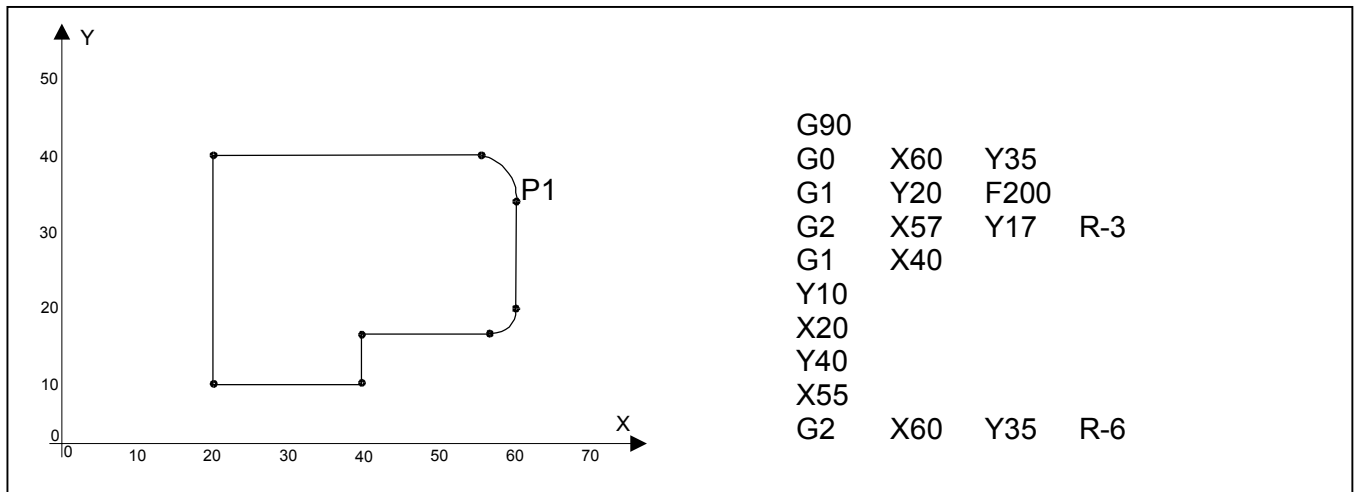
G91 active incremental programming

The absolute system allows the co-ordinates to be programmed referred to the active origin.

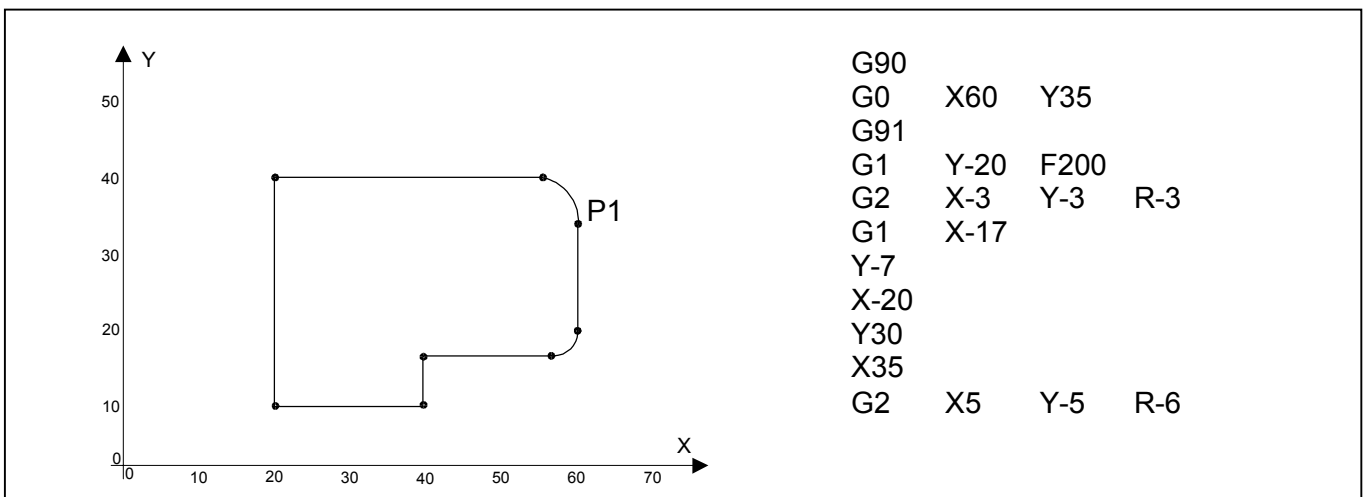
The incremental system allows the programming of the displacements referred to the last position reached by the axes.

The display of the co-ordinates of the axes does not depend on the system selected.

e.g.; programming with absolute co-ordinates (G90)



e.g.; programming with incremental co-ordinates (G91)



## G93 Activate “Tangential Tool Guide”

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G94	
Execution	Queued	
Conditions	Not allowed in the “Positioner” models and in the models with 1 or 2 axes	
Compatible functions	-	
Syntax	G93 [X] [Y] [Z] [W] X,Y,Z,W: axes	<b>(1)</b>

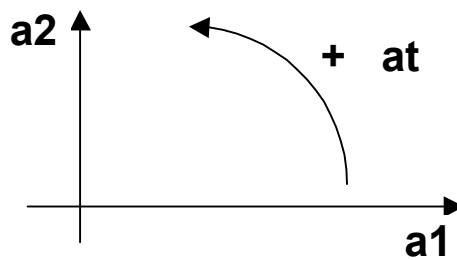
Notes: (1) Must always contain the addresses of the three axes; the first two axes (1st, 2nd) as “axes on the plane of interpolation”, the 3rd axis as tangential. The address W is allowed only if available on the model of controller selected

**Description**

The code G93 is used to select the TANGENTIAL axis (and at the same time define the other two axes on the plane of interpolation) and to activate the automatic guide of the TANGENTIAL TOOL, such that .....  
 G93 <a1> <a2> <at> .....  
 Entering only G93 obtains the default configuration ‘XYZ’.  
 The instruction is modal and can be cancelled by function G94.

e.g.:

G93 XYZ activate the **tangential tool guide** on the z-axis, specify X,Y as axes of the plane of interpolation



**Notes**

The instruction only works if the controller is fitted with the relative option

<b>G94 Deactivate “Tangential Tool Guide”</b>
---

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer / Resident program	
Active	Until the execution of instructions G93	
Execution	Queued	
Conditions	Not allowed in the “Positioner” models and in the models with 1 or 2 axes	
Compatible functions	G93	
Syntax	G94	

**Description**

The code G94 is used to deactivate “automatic guide of the TANGENTIAL axis”.

Example:

```
N...
N5 G94
N...
```

**Notes**

The instruction only works if the controller is fitted with the relative option

## G98 Velocity handling in positioning and interpolation

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host Computer / Resident program	
Active	Until reprogrammed with G98	
Execution	Queued	
Conditions	-	
Compatible functions	-	
Syntax	G98, (n)	

**Description**

The G98 instruction enables the choice of whether the values assigned to the velocity for the current movement performed in interpolation or positioning are independent or interdependent.

The parameter (n) can have the value 0 or 1 with the following meaning.

- 0 = The last velocity specified is valid for subsequent movements both in interpolation or positioning.
- 1 = The last velocity specified in a movement in interpolation is valid for subsequent movements in interpolation. The last velocity specified in a movement in positioning is valid for subsequent movements in positioning.

The instructions for movements in positioning are: G0, G6 and G51.

The instructions for movements in interpolation are: G1, G2, G3 and G33.

The value of the parameter assumed on power up is 0. Therefore the two velocities are interdependent. On power up and until the first explicit assignment is encountered, the value of the parameter F, it means that the velocity will be one half of the maximum velocity specified in the parameters. The moment a G98,1 instruction is encountered, the velocities of positioning and interpolation are equal and correspond to the value assigned. The moment in which the instruction G98,0 is encountered, the two velocities remain independent until the next assignment of the parameter F. From that moment they become equal again.

e.g.: program in which the two velocities are handled in both modes.

<b>G98, 0</b>	; velocities interdependent
<b>G51 X- Y- F5000</b>	; machine zero seeking towards the negative at 5 metres/minute.
<b>G0 X50 Y100</b>	; positioning at 5 metres/minute
<b>G1X200 Y150</b>	; linear interpolation at 5 metres/minute
<b>G98, 1</b>	; velocities independent
<b>G1 X300 Y200 F10000</b>	; linear interpolation at 10 metres/minute
<b>G0 X0 Y0</b>	; positioning at 5 metres/minute





# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 6

-

### Parameter Programming



## CONTENTS

Chapter 6 - Parameter Programming .....	6-1
Introduction .....	6-4
<b>OPERATORS</b> .....	6-4
<b>FUNCTIONS</b> .....	6-8
Buffered Variables .....	6-9
Predefined Variables .....	6-10

## INTRODUCTION

The Q parameters are variables (1 .. 255) to which The values in blocks of assigning calculation variables.

The format is: Qxxx = <expression>

where <expression> can be a numerical value or a mathematical expression whose result will be stored in the variable Q of index xxx.

The <expression> corresponds to that of mathematical type; is formatted by arithmetic operators, functions and operands.

The priority of execution of the mathematical operations in the processing of the <expression> is the following:

( )	brackets
functions	SIN,COS,.....
^	exponent, square root
-	negation
*, /	multiplication, division
+, -	sum, subtraction
logic operators	.AND. .OR. .NOT.

## OPERATORS

The expression must be continuous, there cannot be “spaces” of separation (e.g. Q1=3+6/2).

Every single operation must be separated by at least one space from the rest of the instruction (e.g. Q1=3-6/2 Q2=55\*Q1).

The order of priority of execution of the arithmetical operations in processing the <expression> is the following:

Table summarising The operators				
Group	Symbol	Refer.	Operator Function	Priority
1	+	2Bh	Addition (or positive sign)	1
	-	2Dh	Subtraction (or negative sign)	1
	*	2Ah	Multiplication	2
	/	2Fh	Division	2
2	^	5Eh	Power	3
	SQR	-	Square root	3
3	(	28h	Open brackets	5
	)	29h	Closed brackets	5
4	.AND.	-	Boolean multiplication	2
	.OR.	-	Boolean inclusive sum	2
	.XOR.	-	Boolean exclusive sum	2
	.NOT.	-	Negation	2
5	INT	-	Calculate integer part	3
	=	3Dh	Equals (assignment)	3

The operations with The highest priority are executed first;

Example: Q1=3+6/12\*3-2 gives as The result Q1=2.5  
 Q1=Q1+5+2\*3/5-1 if Q1 initially =2 gives as The result Q1=7.2

**Group 1**

**+ , - , / , \***  
**fundamental operations**

**Syntax:** “± x operation ±y” where:  
 x = 1° operand with its sign  
 y = 2° operand with its sign  
 operation

+	addition
-	subtraction
*	multiplication
/	division

The algebraic sign of a variable is part of The variable itself, therefore expressions of The type 10 – Q1 are accepted, while expressions of The type 0 - -5 are not accepted;

**Examples:**  
**Q1=Q2+Q3      Q1=Q5-3.75**  
**Q3=Q4\*Q2      Q5=3\*Q4-Q2/Q1**

**Group 2**

**^ , SQR**  
**Power , Root**

**Syntax :** “±x ^±y ” where:  
 x = base  
 y = exponent  
 ^ = operation (power/ root)

The exponent can be a variable and can also assume negative values

“SQR(x)” where :  
 SQR = operation square root  
 x = operand (only positive and always between brackets)

**Examples:**  
**Q1=Q5^0.5 (root)    Q1=Q3^2 (squared)**  
**Q1=SQR(Q2^2+Q3^2)**

**Group 3**

**( )**  
**Round brackets**

**Syntax:**  
 round brackets are used as:  
 - Separators for different functions; between brackets is enclosed that which represents the operand of the function;  
 - Indicators of priority for executing the operations according to the arithmetic rules; thus operations enclosed in internal brackets are performed first (respecting always in turn the priorities between the different operations) and then those contained in the external brackets;

A maximum of 10 levels of brackets is allowed.

**Examples:**  
**Q3=3\*(Q5+27)    Q12=(Q5+Q7)-(Q3\*Q6)**

**Group 4**  
**Logic operators**

**.AND.****Is Boolean multiplication;**

With this instruction the value 1 is obtained only if the two operands have the value 1; in all other cases the result of the operation is 0:

1 X 1 = 1  
1 X 0 = 0  
0 X 1 = 0  
0 X 0 = 0

**.OR.****is the Boolean inclusive sum;**

With this instruction the value 1 is obtained whenever one of the terms has the value 1 and the value 0 is obtained only if both terms are 0:

1 + 1 = 1  
1 + 0 = 1  
0 + 1 = 1  
0 + 0 = 0

The operations .AND. and .OR. treat variables considering them as bits.

**Example:**

13b .AND. 11b => 9  
1101 .AND. 1011 => 1001

**.XOR.****is the Boolean exclusive sum;**

with this instruction the value 1 is obtained only when both terms have a different value, in all other cases the result is 0:

1 + 1 = 0  
1 + 0 = 1  
0 + 1 = 1  
0 + 0 = 0

**.NOT.****permits the negation of the binary value of a variable;**

This instruction executes the complement to 1. The decimal value after NOT is therefore a negative value with respect to the previous value less 1.

The instruction NOT applied to a decimal number has no meaning. The decimal part is rounded, after which the instruction is executed.

**Examples:**

Q3=.NOT.Q1 (Q3=1 se Q1=0, Q3=0se Q1<>0)

Q3=Q1.OR.Q2

Q3=Q1.AND.Q2

**Group 5****= , INT****= is the function of assignment;****INT is a function that returns the largest integer less than or equal to the argument (INT(2.5)=2; INT(-2.5)= -2)****Examples:**

Q55=INT(Q12)

Q31=-Q31 change sign of the Q31

Q7=81 assign the value"81" to the variable Q7

Q25=Q25+30 the new value of the variable Q25 will be the sum of the constant "30" plus the value contained in the variable Q25

Q7=QRX assigns the value of the real co-ordinate X to the variable Q7

## FUNCTIONS

Apart from the operators just described, an expression can contain more complex functions indicated by symbols.

The following table lists the functions recognised by the controller

Summary Table of the operators			
Group	Symbol	Operator Function	Priority
6	SIN	Sine	4
	COS	Cosine	4
	TAN	Tangent	4
7	ARS	Arc sine	4
	ARC	Arc cosine	4
	ATN	Arc tangent	4
8	SFL	Set flag ( ' )	5
	TST	Set Reg. Condition ( / )	5

We look at the characteristics of these functions in detail:

### Group 6

**SIN, COS, TAN**  
Direct trigonometrical functions

**Syntax:** "SIN(a)"  
"COS(a)"  
"TAN(a)" where:  
(a) = value of the angle in degrees

The operand in trigonometrical functions is expressed in "decimal degrees 360.000"

**Examples:**  
Q1=SIN(Q3+5)      Q22=COS(Q221)      Q5=TAN(3\*(Q5+Q6))

### group 7

**ARS, ARC, ATN**  
Inverse trigonometrical functions

**Syntax:** "ARS(a)"  
"ARC(a)"  
"ATN(a)" where:  
( a )= value in the range -1..+1

**Examples:**  
Q123=ARS(0.8)      Q12=ARC(Q21-5/(Q8\*Q3))  
Q24=ATN(Q200)

### group 8

**SFL, TST**  
Set flag ( ' ),  
Set Reg. Condition ( / )

**Examples:**  
SFL((QIX.AND.1).OR.(QRD.AND.27))  
TST(Q3)



The parameter functions are allowed in all operations.

The functions must have the operand in brackets: COS(Qxxx), COS(3).

The maximum length of a block is 128 characters with a maximum of 25 variables in the same function. More than one expression of calculation can be introduced in a block of program (max. 10). So strings of the following type are allowed:

```
N1 Q17=Q2*Q3 Q121=SIN(Q17) Q50=COS(Q17)
```

When block "N1" is executed, the programmed variables are loaded one after the other (from left to right).

The sequence of writing establishes the sequence of execution.

La programming of a line can be of a mixed type: conventional (ISO) and PARAMETRIC.

```
N1 G1 X=Q1 Y=Q2
```

## Buffered Variables

The values of the variables between 1...100 "**are not stored**" in memory.

The values of the variables between 101...200 "**are stored**" in memory even after the unit is powered down and/or a new program is selected for execution.

Their eventual initialisation is the responsibility of the <operator> program.

The variables between 201...255 are variables whose meaning is "predefined" (they can be read only (r) or read/write(r/w)).

These variables are identified not only by their number (201..255) but also by symbols.

## Predefined Variables

<b>PREDEFINED Variables</b>				
<b>var.</b>	<b>symbol</b>	<b>Function</b>	<b>notes</b>	<b>format</b>
<b>201</b>	QI1	PLC variable"1" (IN) ---- (variable 80 of the PLC)	r	integer
<b>202</b>	QI2	PLC variable"2" (IN) ---- (variable 82 of the PLC)	r	integer
<b>203</b>	QI3	PLC variable"3" (IN) ---- (variable 84 of the PLC)	r	integer
<b>204</b>	QI4	PLC variable"4" (IN) ---- (variable 86 of the PLC)	r	integer
<b>205</b>	QI5	PLC variable"5" (IN) ---- (variable 88 of the PLC)	r	integer
<b>206</b>	QI6	PLC variable"6" (IN) ---- (variable 90 of the PLC)	r	integer
<b>207</b>	QI7	PLC variable"7" (IN) ---- (variable 92 of the PLC)	r	integer
<b>208</b>	QI8	PLC variable"8" (IN) ---- (variable 94 of the PLC)	r	integer
<b>209</b>	QO1	PLC variable"1" (OUT) ---- (variable 64 of the PLC)	r/w	integer
<b>210</b>	QO2	PLC variable"2" (OUT) ---- (variable 66 of the PLC)	r/w	integer
<b>211</b>	QO3	PLC variable"3" (OUT) ---- (variable 68 of the PLC)	r/w	integer
<b>212</b>	QO4	PLC variable"4" (OUT) ---- (variable 70 of the PLC)	r/w	integer
<b>213</b>	QO5	PLC variable"5" (OUT) ---- (variable 72 of the PLC)	r/w	integer
<b>214</b>	QO6	PLC variable"6" (OUT) ---- (variable 74 of the PLC)	r/w	integer
<b>215</b>	QO7	PLC variable"7" (OUT) ---- (variable 76 of the PLC)	r/w	integer
<b>216</b>	QO8	PLC variable"8" (OUT) ---- (variable 78 of the PLC)	r/w	integer
<b>217</b>	PI	PLC inputs (1...32) (IN) ---- (variable 62 of the PLC)	r	hex
<b>218</b>	PO	PLC outputs (1...32) (OUT) ---- (variable 60 of the PLC)	r/w	hex
<b>219</b>	QRX	real co-ordinate X (absolute)	r	decim.
<b>220</b>	QRY	real co-ordinate Y (absolute)	r	decim.
<b>221</b>	QRZ	real co-ordinate Z (absolute)	r	decim.
<b>222</b>	QTX	theoretical co-ordinate X (absolute)	r	decim.
<b>223</b>	QTY	theoretical co-ordinate Y (absolute)	r	decim.
<b>224</b>	QTZ	theoretical co-ordinate Z (absolute)	r	decim.

PREDEFINED Variables				
var.	symbol	function	note	format
225	QEX	follow error of X	r	decim.
226	QEY	follow error of Y	r	decim.
227	QEZ	follow error of Z	r	decim.
228	QIX	X inputs	r	hex
229	QIY	Y inputs	r	hex
230	QIZ	Z inputs	r	hex
231	QOX	X outputs	r	hex
232	QOY	Y outputs	r	hex
233	QOZ	Z outputs	r	hex
234	QRD	Machine status register "D"	r	hex
235	QRE	Alarm register "E"	r	hex
236	QR\	Condition register "\"	r	hex
237	QVX	X-axis velocity	r	decim.
238	QVY	Y-axis velocity	r	decim.
239	QVZ	Z-axis velocity	r	decim.
240	QSY	External pulse counter	r/w	integer
241	QFD	Read FEED	r	integer
242	QPT	Point velocity (in interpolation)	r	decim
243	QTP	variable time in msec. (resolution = 2 msec)	r/w	decim
244	QPN	image of the control panel	r	hex
245	QAX	Co-ordinate of the X probe	r	decim
246	QAY	Co-ordinate of the Y probe	r	decim
247	QAZ	Co-ordinate of the Z probe	r	decim
248	QAW	Co-ordinate of the W probe	r	decim
249	QRW	real co-ordinate of W (absolute)	r	decim
250	QTW	theoretical co-ordinate of W (absolute)	r	decim
251	QEW	follow error of W	r	decim.
252	QIW	W inputs	r	hex
253	QOW	W outputs	r	hex
254	QVW	W-axis velocity	r	decim.
255	QNR	Controller No.	r	decim.

The variables can be read using the instruction **%81**.

The value contained in the variable will be transmitted in the format specified in the "format" column, i.e. as an integer, decimal or hexadecimal.





# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 7

-

### “&” Instructions - Special Functions



# CONTENTS

Chapter 7 - “&” Instructions - Special Functions .....	7-1
CONTENTS .....	7-3
Introduction .....	7-4
&00 Select <PROG> mode .....	7-5
&01 Select <AUTO> mode .....	7-6
&02 Select <ON-LINE> mode .....	7-7
&03 Select <JOG> mode .....	7-8
&04 List of a Stored Program .....	7-9
&05 List of Stored Programs .....	7-10
&06 Cancel Program .....	7-11
&07 Load Default Parameters .....	7-12
&08 Select Mode of Communication .....	7-13
&09 Select Language of Error messages .....	7-14
&10 - &11 Enable - Disable External Panel .....	7-15
&12 Define Number of the Controller.....	7-16
&15 Modify the Communication Protocol.....	7-17
&20 Write VARIABLES “Q 1...255” .....	7-18
&30 Modify the Velocity of the Axes (in percentage) .....	7-19
&31 Modify the Analogue Voltage (defined with @52).....	7-20
&70 - &71 - &79 - &80 - &84 - &88 .....	7-21
&51 Stop Recording Co-ordinates .....	7-22
&52 Start Recording Co-ordinates.....	7-23
Procedure for recording the co-ordinates .....	7-25
Packets of recording data .....	7-25
Packet of end of recording.....	7-28
Practical advice .....	7-29

## **INTRODUCTION**

These are instructions that the controller uses to perform special functions, such as:

- Select the mode of operation
- Manage programs (list, catalogue, cancel)
- preset default parameters
- define execution modes
- enable external control panels
- immediate instructions executed from within the program

### **Note**

If not otherwise specified, the examples of communication supplied together with the descriptions of the instructions always refer to communication mode of “computer without echo” and the “on-line” mode of operation.



## &00 Select <PROG> mode

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate; from program	
Conditions	External panel disabled; no program already being loaded	
Syntax	&00 [, (p)] (p): number of program to load range : 1..136; max. 3 ASCII-decimal digits	<b>(1)</b>

Notes: (1) programming &00 without parameters, program No.1 is selected

**Description**

Used to select the mode of operation PROG and enable the loading of the program (p) specified in the instruction.

e.g.: select PROG and enable loading of the program 32.

```

PC          > <cr>
CONTROLLER > '$'
PC          > '&00,32' <cr>
CONTROLLER > '~'
    
```

note the change of the PROMPT character

## &01 Select <AUTO> mode

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	External panel disabled	
Syntax	&01 [, (p)] (p): number of program to EXECUTE  range : 1..136; max. 3 ASCII-decimal digits	<b>(1)</b>

Notes: (1) If the No. of the program is omitted, program No.1 is selected  
The parameter (t) is used in handling the tasks

**Description**

Used to select the mode of operation AUTO and enable the execution of the program (p) specified in the function. The execution of the program will begin on the START instruction.

e.g.: select AUTO mode

Select AUTO and enable execution of the program 10: note the change of the PROMPT character.

```
PC           > <cr>
CONTROLLER  > '$'
PC           > '&01,10' <cr>
CONTROLLER  > '|'
```

The program 27 is not present

```
PC           > '&1,27' <cr>
CONTROLLER  > '?0-c02: Program not present in memory'
CONTROLLER  > '$'
```

## &02 Select <ON-LINE> mode

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	External panel disabled	
Syntax	&02	<b>(1)</b>

Notes: (1) is an instruction without parameters

**Description**

Used to select the ON-LINE mode of operation.

e.g.: select ON-LINE mode

```

PC                > <cr>
CONTROLLER        > '~'
PC                > '&02' <cr>
CONTROLLER        > '$'           note the change of the PROMPT character.
    
```

## &03 Select <JOG> mode

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	External panel disabled	
Syntax	&03	<b>(1)</b>

Notes: (1) is an instruction without parameters

**Description**

Used to select the mode of operation JOG.

e.g.: select JOG

PC	>	<cr>	
CONTROLLER	>	'\$'	
PC	>	'&02' <cr>	
CONTROLLER	>	'_'	note the change of the PROMPT character.

## &04 List of a Stored Program

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	Program present	
Syntax	&04 , (p) (p) : number of program to send	<b>(1)</b>

Notes: (1) range 1..136: max. 3 ASCII-decimal digits

**Description**

Activates the sending, over the communication link, of the program (p) specified in the instruction. The sending is executed line by line: at every <cr> sent subsequent to the instruction, the controller will respond by sending a line of program. Use the instruction ^P <dle> to abort the listing.

## &05 List of Stored Programs

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&05	<b>(1)</b>

Notes: (1) is an instruction without parameters

**Description**

Activates the sending, over the communication link, of the lines of TAG of all the programs stored on the controller.

The sending is executed line per line: at every <cr> sent subsequent to the instruction, the controller will reply by sending the number of the program followed by the corresponding line of TAG.

Use the instruction ^P to abort the listing

e.g.: list of programs stored

On the controller the programs 10, 27 and 105 are present

```

PC           > '&05' <cr>
CONTROLLER  > '~'
PC           > <cr>
CONTROLLER  > '010 : "TAG PROGRAM 10"
CONTROLLER  > '~'
PC           > <cr>
CONTROLLER  > '027 : "TAG PRG 27"
CONTROLLER  > '~'
PC           > <cr>
CONTROLLER  > '105 : "Progr. 105"
CONTROLLER  > '~'
PC           > <cr>
CONTROLLER  > '~'
    
```

## &06 Cancel Program

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	“PROG” mode, no program already being loaded	
Syntax	&06, (p) (p): number of program to cancel	<b>(1)</b>

Notes: (1) value in the range 1..136 : max. 3 ASCII-decimal digits

**Description**

Cancel the program (p) specified in the function.

e.g.: cancel program

```

PC          > <cr>
CONTROLLER > '~'
PC          > '&06,18' <cr>      Cancel program 18
CONTROLLER > '~'
    
```

## &07 Load Default Parameters

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	“ON-LINE” mode, no instruction in execution: the axes must be stopped and there must not be any instruction with queued execution waiting to be executed	
Syntax	&07, (n) (n): operation to execute	<b>(1)</b>

Notes: (1) value in the range 0..3: max. 1 ASCII-decimal digit

**Description**

The instruction is used to load the default values of all the parameters programmable with the functions of customisation ('@' functions) and/or of cancelling in block all the programs stored and/or of cancelling the program loaded in the PLC.

The letter (n) can have the following values:

'0' = load CNC default parameters

'1' = cancel all the stored CNC programs

'2' = load the CNC default parameters and cancel all the stored CNC programs.

'3' = cancel the program loaded in the PLC

e.g.: load default parameters

No instruction in execution

```
PC > '&07,0' <cr>
CONTROLLER > '$'
```

**Important**

Use with caution: after using this function, all '@' functions required to customise the system must be reprogrammed.



## &08 Select Mode of Communication

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&08, (m) (m): mode of communication code	<b>(1)</b>

Notes: (1) value in the range 1..3: max. 1 ASCII-decimal digit

**Description**

Defines the mode of communication of the controller, according to the number attributed to (m) :

- '1' = terminal
- '2' = computer without echo
- '3' = computer with echo

e.g.: select mode of communication

the “computer without echo” mode is active:

```

PC          > '&08,3' <cr>          enable “computer with echo” mode
CONTROLLER > '$'
PC          > 'G0 X123' <cr>
CONTROLLER > 'G0 X123' <cr>
CONTROLLER > '$'
    
```

## &09 Select Language of Error messages

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&09, (m) (m): code of language to select	<b>(1)</b>

Notes: (1) value in the range 1..4: max. 1 ASCII-decimal digit

**Description**

Defines the language for the error messages, according to the value attributed to (m):

- '1' = Italian (default)
- '2' = English
- '3' = French
- '4' = German

e.g.: select language for error messages

PC > '&09,2' <cr>

## &10 - &11 Enable - Disable External Panel

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&10 &11	(1) (1)

Notes: (1) is an instruction without parameters

**Description**

The function &10 enables the use of the external panel.

On power-up of after a software reset of a controller system that provides a remote control panel, this latter can be enabled automatically with the instruction @97.

With &11 the external panels are disabled.

With this condition active, the selection of the mode of operation is possible only with the functions &00, &01, &02 and &03.

e.g.: enable external panel

```
PC > '&10' <cr>
CONTROLLER > '$'
PC > '&00,18' <cr>
CONTROLLER > '?0-c08: Instruction not executable'
CONTROLLER > '$'
```

e.g.: disable external panel

```
PC > '&11' <cr>
CONTROLLER > '$'
PC > '&00,18' <cr>
CONTROLLER > '~'
```

**Important**

The active state of the panel on power-up is programmed with parameter @97.

## &12 Define Number of the Controller

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&12 , (n) (n) : number of the controller; value in the range 0..99	

**Description**

This instruction, that is used to define the number of the controller, is required in systems that have more than one controller connected on the same RS422 or RS485 serial link with the P.C.

In these systems, it is necessary to identify each controller with a different number so that any particular instruction to hook-up communication connects with the correct controller. For this topic, refer to the immediate instruction: Ctrl-G hook-up communication.

The instruction &12 is always accepted by all the controllers on the RS232 serial link.

e.g.: define number of the controller

```
PC           >   '&12,3' <cr>
CONTROLLER  >   '$'
```

**Important**

The controllers are supplied configured with the number 0. Thus, when there are more than one controller on a single RS422 or RS485 serial link, each controller must be configured with its own number, connecting it on its own to the P.C. and using the “hook-up” instruction ^G^G.

If on power-up an error is detected in the memory of the parameters in the e<sup>2</sup>prom, the controller automatically assumes the number 0.

The controller, nevertheless, is able to be “re-hooked” (see the instruction ^G^G) and reconfigured with the correct number.

## &15 Modify the Communication Protocol

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&15 , (t) (t) : number in the range 0..23	

**Description**

This instruction is used to modify the protocol of communication between the PC and the controller. On power-up, the system is set up to communicate with the following serial protocol :

```

baud-rate    = 9600
data-bits    = 8
parity       = ODD
stop-bits    = 1
    
```

With this instruction it is possible to modify the BAUD-RATE and the PARITY .  
The parameter (t) is a number (0-23) with the following values :

$t = bd + prt * 8$       in which :

- bd =0 -> 300 baud
- bd =1 -> 600 baud
- bd =2 -> 1200 baud
- bd =3 -> 2400 baud
- bd =4 -> 4800 baud
- bd =5 -> 9600 baud
- bd =6 -> 19200 baud
- bd =7 -> 38400 baud

- prt =0 -> EVEN
- prt =1 -> ODD
- prt =2 -> NO PARITY

e.g.: modify the communication protocol

```

PC          > '&15,22' <cr>          select the protocol 19200,8,1,NO-PARITY
CONTROLLER > '$'
    
```

## &20 Write VARIABLES “Q 1...255”

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&20 Q(m) = value (m) : number of the variable that is to be modified; range 1..255 “value” : data to enter for the variable	<b>(1)</b>

Notes: (1) e.g.: Q10 = 123 the variable Q10 is made equal to 123

**Description**

This instruction is used to modify the contents of the INTERFACE variables Q 1..255 also during the execution of a program (Automatic) in a way that can affect the behaviour of the program in execution.

e.g.: write variable

```

PC          > '&20 Q55=123' <cr>
CONTROLLER > '$'
```

## &30 Modify the Velocity of the Axes (in percentage)

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&30, (p) (p) : number in the range -99..100	

**Description**

This instruction is used to modify the velocity of the axes during their movement in an allowed range from 1 % to double the nominal velocity.

This function is only active after being enabled with the instruction @80.

Also all the subsequent velocity settings will undergo the “modification” of the velocity until disabled with the instruction @80 or on programming the value (p) =0 that sets the nominal velocity.

The parameter (p) is a number ( -99 .... 100) that represents how much the velocity of the axes must be modified.

- (p) = -99 : velocity = 1/100 of that programmed
- (p) = 100 : velocity = double of that programmed
- (p) = 0 : velocity = programmed velocity

e.g.: modify velocity

```
PC > '&30,21' <cr>
CONTROLLER > '$'
```

## &31 Modify the Analogue Voltage (defined with @52)

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	
Syntax	&31, (m) (m) : number in the range -99..100	

**Description**

This instruction is used to modify the output voltage (velocity of the spindle) during the work in an allowed range from 1 % to double that of the nominal velocity.

Also all the subsequent velocity settings will undergo the "modification" of the velocity until disabled with the instruction @80 or on programming the value (p) =0 that sets the nominal velocity.

The parameter (m) is a number ( -99 .... 100) that represents how much the velocity of the axes must be modified.

- (m) = -99 : velocity = 1/100 of that programmed
- (m) = 100 : velocity = double of that programmed
- (m) = 0 : velocity = programmed velocity

e.g.: modify analogue voltage

```
PC > '&31,21' <cr>
CONTROLLER > '$'
```



**&70 - &71 - &79 - &80 - &84 - &88**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, memory of program (execution programs)	
Active	In the block it is programmed	
Execution	Immediate (from program)	
Conditions	Depends on the active mode: ON-LINE = not allowed PROG = allowed AUTO = allowed only from prog.memory JOG = allowed only from prog.memory	
Syntax	&70 &71 &79 &80 &84 &88	(1) (1) (1) (1) (1) (1)

Notes: (1) is a function without parameters

**Description**

- &70 STOP CNC
- &71 START CNC
- &79 SET EMERGENCY
- &80 RESET EMERGENCY
- &84 ANNUL INSTRUCTION IN EXECUTION
- &88 SINGLE STEP

These are instructions that duplicate certain functions that can be executed using immediate instructions, making it possible to execute them from within a program.

e.g.:

Send the instruction during the loading of a program

```
PC > '&84' <cr>
CONTROLLER > '+'
CONTROLLER > '|'
```

Send the instruction during the execution of a program

```
PC > '&70' <cr>
CONTROLLER > '|'
```

## &51 Stop Recording Co-ordinates

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the block it is programmed	
Execution	Immediate	
Conditions	-	<b>(1)</b>
Syntax	&51	<b>(2)</b>

- Notes:
- (1) if the recording is not active the function has no effect
  - (2) is an instruction without parameters

**Description**

The function &51 is used to end the procedure of recording the theoretical and actual co-ordinates. It is always executed, whatever the mode of working of the controller and the type of stop selected with the function &52.

e.g.: stop recording co-ordinates

```
PC           >   '&51' <cr>
CONTROLLER  >   '$'
```

**Important**

The end of the procedure of recording is indicated by the controller sending a packet of particular recordings (see paragraph at the end of this chapter) made up of only 2 bytes. This "end of recording" packet is transmitted **before the PROMPT** sent in response to the instruction &51.

## &52 Start Recording Co-ordinates

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer	
Active	In the programmed block	
Execution	Immediate	
Conditions	ON-LINE or AUTO Mode if the recording is already active, an error is signalled	
Syntax	&52 , (axes) , (period) , (stop) [X] [Y] [Z] [W] (axes) : axes to be recorded (period) : sampling period (stop) : type of stop	(1) (2)

- Notes:
- (1) The addresses Y, Z and W are allowed only if available on the model of controller chosen
  - (2) At least one address must be defined

**Description**

The &52 function is used to begin the recording of the co-ordinates of one or more axes with two modes of reply.

Mode 0: Recording of the variations between the theoretical co-ordinates of the axes selected and the actual values (default value on power up)

Mode 1: Recording of the actual values of the co-ordinates for the chosen axes.

At the moment it is possible to activate mode 1 only by using the G74 instruction for activating the LASER reading of the co-ordinates.

The recording can be activated only if the controller is in ON-LINE or AUTO.

The procedure is set up with the following parameters:

- AXES defines for which axes the recording must be performed. At least one axis must be specified.
- PERIOD defines the period for sampling and sending the data to be recorded or select the sending of data on every variation of the actual co-ordinates of any one of the specified axes.  
È a value in the range 0..9 with the following meaning:  
'0' : send data at every variation between the theoretical or actual co-ordinate  
'1' : sample co-ordinates every 4 ms  
'2' : sample co-ordinates every 8 ms  
'3' : sample co-ordinates every 12 ms  
'4' : sample co-ordinates every 16 ms  
'5' : sample co-ordinates every 20 ms  
'6' : sample co-ordinates every 24 ms  
'7' : sample co-ordinates every 28 ms  
'8' : sample co-ordinates every 32 ms  
'9' : sample co-ordinates every 36 ms
- STOP it is possible to define if the recording will stop only on instruction (see &51), or will take place automatically at the end of the execution of the program selected in the AUTO mode.  
È a value in the range 0..1 with the following meaning:  
'0' : stop only on a &51 instruction (obligatory if in the ON-LINE mode)  
'1' : stop at the end of the execution program or on a &51 instruction.

It will be noticed that the instruction to start recording can be sent before or after the instruction to start the execution of the program (Ctrl-C or push-button on the remote panel).

e.g.:

start recording on the axes X, Y and Z in ONLINE  
 Period: 8 msec.  
 Stop: on command

```
PC                >    '&52,XYZ,2,0' <cr>
CONTROLLER        >    '$'
```

Start recording only on the Y-axis in AUTO  
 Period: variation co-ordinate  
 Stop: at the end of the execution of the program

```
PC                >    '&52,Y,0,1' <cr>
CONTROLLER        >    '|'
```

### Important

The controller begins to send the recording data **after sending the PROMPT**, in reply to the &52 instruction. So it will happen only if the instruction received is formally correct.

The recording data, of which a complete description is supplied at the end of this chapter, is sent continually by the controller in packets of 5, 8, 11 or 14 bytes (depending on the number of axes to record) until a stop condition is verified (by instruction or at end of program).

During this phase, the controller continues to communicate regularly, handling the eventual instructions that will be sent to it over the communication channel: whenever the instructions are “read instructions” (i.e instructions that require a reply from the controller) it is possible that one or more packets of recording data will be sent “within” the reply expected by the request.

The bytes transmitted can assume any value between 0 and 255. Therefore it is obligatory to program the serial link (see &15) in a way that functions with **8 data bits** before launching the recording.

## PROCEDURE FOR RECORDING THE CO-ORDINATES

The recording procedure, launched by the &52 instruction, consists in transmitting a series of **packets of data** to the controller, during the execution of one or more instructions in the ON-LINE or of a program in AUTO, containing the information concerning:

- **variations between the actual and theoretical co-ordinates** of one or more axes (mode 0),
- **absolute values of the actual co-ordinates** of one or more axes (mode 1).

The sending of the packets is handled automatically by the controller, until the procedure is concluded with the dedicated &51 instruction or automatically on conclusion of the program.

### Packets of recording data

#### Period of sampling and sending

During the recording procedure, the theoretical and actual co-ordinates are samples at the moment of a variation or periodically at an interval between 4 and 36 ms., as specified in the start recording instruction (see instruction &52).

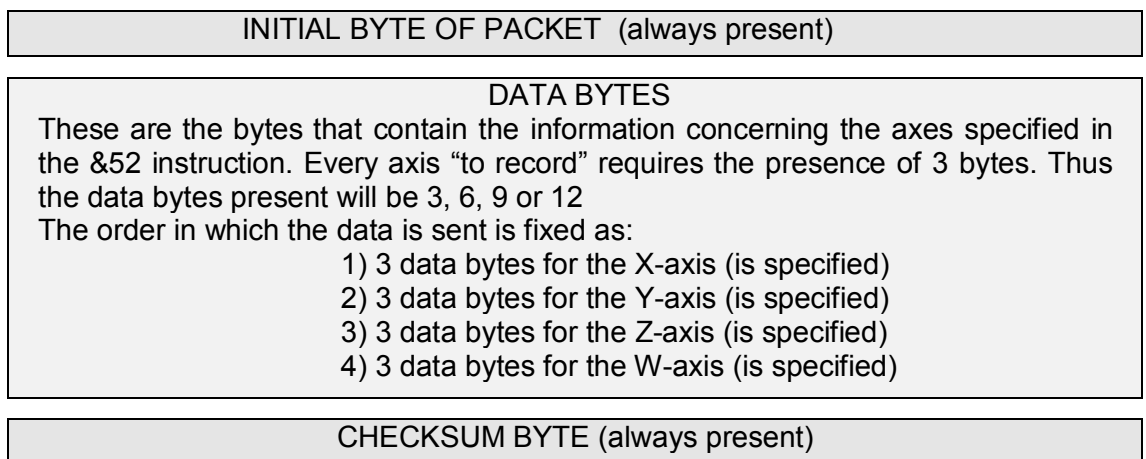
At every sampling, the controller memorises the variations of the co-ordinates and sends them automatically to the communication channel without waiting for any request from the host. Thus the period of sending a packet of recording data will vary as a function of the movements of the system or equal to the sampling period.

In the event that it is selected to send on every variation of a co-ordinate, the time between the sending of one packet and the next will never be less than 4ms.

#### Format of the packet of data

The packet of recording data is always made up of **binary bytes** with values in the range 0 .. 255. For this reason, it is **obligatory** that the serial link is configured to function with **8 data bits**.

The structure of the packet is the following:



The length of the packet varies with the number of axes specified for the recording:

- 1 axis: 5 bytes
- 2 axes: 8 bytes
- 3 axes: 11 bytes
- 4 axes: 14 bytes

**Initial byte of packet**

7	6	5	4	3	2	1	0
1	e	0	y	x	x	x	x

The initial byte of the packet is normally made up, as a function of the number of axes selected for recording, as one of the following values:

	<b>Mod. 0</b>	<b>Mod. 1</b>
1 axis	Hex 84	Hex 94
2 axes	Hex 87	Hex 97
3 axes	Hex 8A	Hex 9A
4 axes	Hex 8D	Hex 9D

In practice the least significant “nibble” (xxxx) indicates the number of bytes that will follow that of the initial one, including the checksum: 4, 7, 10 or 13.

Bit 4(y) signals whether the information is to be considered relative to the variations of the theoretical and actual co-ordinates or relative to the absolute actual position.

Bit 6(e), if set to 1, indicates that one or more packets of data have been lost because the host has not been stimulated sufficiently to unload the transmission FIFO. In this case the least significant nibble contains the value 1, because the data of the co-ordinates will not be transmitted, but only the checksum.

**Data bytes**

**Mode 0: increments of the theoretical and actual co-ordinates**

The length of the packet is therefore variable as a function of the number of axes chosen for the recording:

- 1 axis : 5 bytes
- 2 axes : 8 bytes
- 3 axes : 11 bytes

**Byte at beginning of packet**

7	6	5	4	3	2	1	0
1	0	0	0	x	x	x	x

The byte at the beginning of the packet is always made up, as a function of the number of axes chosen for the recording, of one of the following values:

- 1 axis : 84 Hex
- 2 axes : 87 Hex

3 axes : 8A Hex

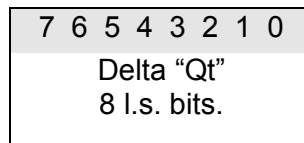
In practice the less significant "nibble" (xxxx) indicates the number of bytes that will follow the initial one, including the checksum: 4, 7 or 10.

**Data bytes**

The following description regards the block of 3 bytes that contain the variation of the theoretical and actual co-ordinates of an axis and is therefore valid for each chosen axis.

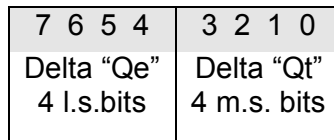
**- First byte:**

Contains the 8 less significant bits of the variation of the theoretical co-ordinate.



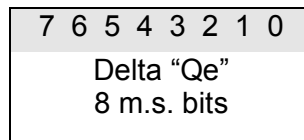
**- Second byte:**

The l.s. nibble contains the most significant 4 bits of the variation of the theoretical co-ordinate, while the most significant nibble contains the least significant 4 bits of the variation of the actual co-ordinate.



**- Third byte:**

Contains the 8 most significant bits of the variation of the actual co-ordinate.

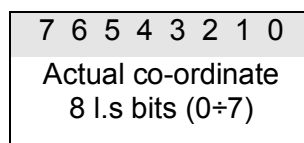


**Mode 1: actual absolute co-ordinates**

The following description refers to the block of 3 bytes containing the actual co-ordinates of an axis and thus valid for each of the selected axes.

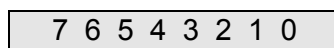
**- First byte:**

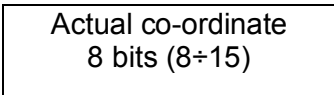
Contains the 8 least significant bits of the actual co-ordinates.



**- Second byte:**

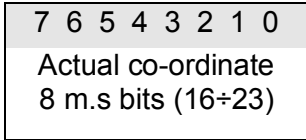
Contains the 8 intermediate bits of the actual co-ordinates.





**- Third byte:**

Contains the 8 most significant bits of the actual co-ordinates.



**Checksum Byte**

This is always the last byte, sent to enable the verification of the correct reception of all the component bytes in the packet of recording data.

It is calculated **complementing to 1 the arithmetic sum** of all the previous bytes, including the byte at the beginning of the packet.

**Packet of end of recording**

The packet of end of recording, sent on the instruction to stop or at the end of the execution of the program (see &50), is in practice a particular data packet, of only 2 bytes, in which the information regarding the variations of the co-ordinate are missing. Its format is the following:



The packet of end of recording is also used to indicate any eventual error of **transmission FIFO full** that can be generated during the sending of the packets of recording. In this case the values sent to close the recording are:



The eventual indication of this error means that the data received from the controller are not usable, as they are incomplete.



## Practical advice

With the purpose of being able to implement the procedure of recording correctly, certain details regarding the communication speed and the maximum motor or encoder frequency allowed during the execution of the “recorded” movements must be checked.

### Communication speed

**Be careful when selecting the baud-rate** in order to avoid filling the transmission FIFO.

The choice is influenced by two factors:

- 1) the number of axes "to record": on this depends the number of bytes that make up the packet and consequently the time of transmission of the whole packet, given a fixed baud-rate.
- 2) the time of sampling and of transmission: such time must be comfortably greater than that required to transmit a packet of recording data.

The following table shows the values of **minimum baud rate** that can be used as a function of the sampling time and the number of axes. The values are calculated on the basis of a protocol “1 start bit, 8 data bits, NO PARITY, 1 stop bit”. It also takes account of at least one more bit in the interval between one byte and the next.

Sampling time	1 axis (5 bytes)	2 axes (8 bytes)	3 axes (11 bytes)	4 axes (14 bytes)
delta Q.	19200	38400	38400	38400
4 ms	19200	38400	38400	38400
8 ms	9600	19200	19200	38400
12 ms	4800	9600	19200	19200
16 ms	4800	9600	9600	19200
20 ms	4800	4800	9600	9600
24 ms	2400	4800	9600	9600
28 ms	2400	4800	4800	9600
32 ms	2400	4800	4800	9600
36 ms	2400	4800	4800	4800

Similarly, if the Picasso 2000 controller is used, it is necessary to ensure that the PC bus can read all the bytes that make up the packet of recording data in a time that is less than that of the chosen sampling time.

**Maximum encoder or motor frequency**

When reading the co-ordinates in mode 0, the “recorded” movements, that is those executed while the recording procedure is active, have the following limit: **the encoder frequency (d.c. motor axes) or motor frequency (stepper motor axes) must be such as to not cause an overflow of the 12 bits dedicated to memorising the variation of the co-ordinate.**

This, being allowed, in the chosen sampling time, a maximum variation of co-ordinate of 2047 steps or pulses (absolute value), the maximum values of frequency allowed as a function of the sampling time will be the following:

Sampling time	Maximum frequency (Hz.)
delta Q.	511750
4 ms	511750
8 ms	255875
12 ms	170583
16 ms	127937
20 ms	102350
24 ms	85291
28 ms	73107
32 ms	63968
36 ms	56861



# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 8

-

### “M” Instructions



## CONTENTS

Chapter 8 - "M" Instructions .....	8-1
Introduction .....	8-4
M00 Stop Program .....	8-5
M02 End of Main Program.....	8-6
M10 ... M41 request the execution of a PLC routine .....	8-7
M99 End of Subprogram .....	8-8

## **INTRODUCTION**

These are instructions used in programs to perform the following functions:

- stop program (M00)
- end main program (M02)
- end subprogram (M99)
- request execution of a routine PLC (M10 → M41)

# M00 Stop Program

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program: queued	
Admissibility	Conditioned	
Conditions	Depends on the active mode: ONLINE = not allowed PROG = allowed inside the main program or in a subprogram AUTO = allowed JOG = not allowed	
Syntax	M00	<b>(1)</b>

Note: (1) is an instruction without parameters

**Description**

The execution of this instruction causes the program to stop and await a CNC start instruction. The instruction of start could be sent on the communication link (serial – bus) using the immediate instruction Ctrl-C or, in systems that contain the external control panel, by pressing the START button.

e.g.: at the end of the execution of the arc of a circle (G02) the program stops, awaiting a new “start”

```

&00,32           ; select <PROG> mode
[
"Program Title"
G01 X0 Y0 F1000
G02 X123 Y345 R500
M00             ; Stop program
G0 X500 Y500
G01 X0 Y0
M02
]
    
```

## M02 End of Main Program

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program: queued	
Admissibility	Conditioned	
Conditions	Depends on the active mode: ONLINE = not allowed PROG = allowed in the main program AUTO = allowed JOG = not allowed	
Syntax	M02	<b>(1)</b>

Note: (1) is an instruction without parameters

**Description**

In the programming phase, this instruction is sent to close the “Main Program”.  
 Eventual subprograms must be sent after the M02 instruction.  
 In execution, M02 indicates the end of the program to the controller.

Example:

```

&00,32                ; select <PROG> mode
[
"Program Title"
G01 X0 Y0 F1000
G02 X123 Y345 R500
G01 X0 Y0
M02                  ; End of program
]
    
```



**M10 ... M41 request the execution of a PLC routine**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program: queued	
Admissibility	Conditioned	
Conditions	Depends on the active mode: ONLINE = allowed PROG = allowed inside the main program or in a subprogram AUTO = allowed JOG = allowed	
Syntax	M..	<b>(1)</b>

Note: (1) is an instruction without parameters  
 it is only allowed for controllers with a Plc

**Description**

Requests the execution of a PLC cycle and waits for it to finish.  
 The PLC must indicate the end of the cycle by resetting the corresponding point to the M instruction, in order to give back the control to the CNC.

Example:

```

&00,32                ; select <PROG> mode
[
"Program Title"
G01 X0 Y0 F1000
G02 X123 Y345 R500
M21                    ; Await synchronism from PLC
G0 X500 Y500
G01 X0 Y0
M02
]
    
```

# M99 End of Subprogram

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program: instantaneous	
Admissibility	Conditioned	
Conditions	Depends on the active mode: ONLINE = not allowed PROG = allowed in a subprogram AUTO = allowed JOG = not allowed	
Syntax	M99	<b>(1)</b>

Note: (1) is an instruction without parameters

**Description**

In the programming phase, this instruction is sent to “close” a subprogram. In execution, M99 indicates the end of the subprogram to the controller, causing it to continue with the next instruction following that of the “call” (no repetition) or at the first instruction of the same subprogram (repetition of the subprogram). See also functions G30, G31 and G32.

Example:

```

&00,32                ; select <PROG> mode
[
"Program Title"
G0 X12 Y5 F2000
G30 L10                ; Recall subprogram no.10
G00 X100
M02
:10
G04 F3.5
M99                    ; End of subprogram
]
    
```



# CNC

## PROGRAMMING MANUAL FOR S&h CONTROLLERS

### Chapter 9

-

### Program Handling Instructions



## CONTENTS

Chapter 9 - Program Handling Instructions .....	9-1
Introduction .....	9-4
[ Beginning of program .....	9-5
"Program Tag " .....	9-6
# Define Label .....	9-7
: Beginning of subprogram .....	9-8
] End of Program .....	9-9

## **INTRODUCTION**

These are special instructions that control the sending of a program, enable the insertion of labels and of subprograms, and finally make it possible to compile the program and store it in the non-volatile memory of the controller.

However they are instructions that are accepted in the PROG mode of operation, and during the execution of the programs (AUTO and JOG modes) they are considered as void instructions. They are not allowed in the ON-LINE mode of operation.

<b>[ Beginning of program</b>
-------------------------------

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program	
Conditions	It must be the first instruction of a program, that is the first instruction sent after having selected the number of the program to load with the &00 instruction. Once sent, it is no longer allowed.	
Syntax	[	<b>(1)</b>

Note: (1) it is an instruction without parameters

**Description**

It must be the first instruction of a program, that is the first instruction sent after having selected the number of the program to load with the &00 instruction (see related & function). Once sent, it is no longer allowed.

e.g.: beginning of program

```

&00,32           ; select <PROG> mode, program No. 32
[                Beginning of program
"Program Title"
.....
.....
.....
]
```

**Important**

After the ‘ [ ‘ instruction, it is obligatory to send the instruction for defining the title of the program.

**“Program Tag “**

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program	
Conditions	This must be the second instruction of a program, that is the first instruction following the instruction to begin loading ' [ '. Once sent, it is no longer allowed	
Syntax	“(string)” (string) : string of the TITLE; The ASCII string of the “Tag” must be enclosed between two “ “ characters “	<b>(1)</b>

Note: (1) The string must be made up of a maximum of 16 ASCII characters

**Description**

This instruction enables the insertion of a string of ASCII characters at the head of the program, denominated “Tag” that constitutes the “title” of the program.  
This string will be that sent by the controller during the execution of the instruction &O5 (list of stored programs).

Example:

```

&O0,5           ; select <PROG> mode, program No. 5
[
"Program 5 "    | Title of the program
.....
.....
.....
]
    
```

**Important**

After defining the “tag”, any of the instructions ('G', 'M', '@' and '&' functions) that make up the “Main Program”, can be sent.  
Remember that the controller never accepts as “input” the characters that are used for the PROMPTS: '\$', '~', '\_ e '|'.  
.



<b># Define Label</b>
-----------------------

Pic2000	Goya	Rubens						
---------	------	--------	--	--	--	--	--	--

**properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program	
Conditions	Can only be sent inside the main program or in a subprogram	
Syntax	# (label) (label) : numerical value of the label	<b>(1)</b>

Note: (1) value in the range 1..9999: max. 4 ASCII-decimal digits

**Description**

This is used to define a numerical “label” which is used as a reference with “jump” instructions (functions G20, G21 and G22).

e.g.: define Label

The program remains in “loop” awaiting the sending of the code <SUB>

```

&00,21           ; select <PROG> mode
[
"Program Title"
#100             --
G69 O1          -- Program example --
G21 L100        --
    
```

**: Beginning of subprogram**

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program	
Conditions	Can only be sent after the main program has been closed with an M02 instruction or after the closure of another subprogram with the M99 instruction	
Syntax	: (subprogram) (subprogram) : numerical value of the subprogram	<b>(1)</b>

Note: (1) Value in the range 1..9999: max. 4 ASCII-decimal digits

**Description**

Identifies the beginning of a subprogram by defining a numerical value to use as a reference when using the “call” instructions (functions G30, G31 and G32).

e.g.: beginning of subprogram

The Main Program uses the call to subprogram 10 for executing the Set-Point on the X- and Y-axes. At the end of the set-point routine, the program returns from the subprogram and continues with the next instruction following the call.

```

[
"Program Title"
G0 X0
G30 L10
G0X100F1000
M02

:10
G51 X+ Y-
M99
]
```

# ] End of Program

Pic2000	Goya	Rubens					
---------	------	--------	--	--	--	--	--

**Properties**

Source	Host computer, program memory	
Active	In the block it is programmed	
Execution	From program	
Conditions	It can only be sent after the main program has been closed with an M02 instruction or after the closure of another subprogram using the M99 instruction.	
Syntax	]	<b>(1)</b>

Note: (1) it is an instruction without parameters

**Description**

This is the instruction that concludes the insertion of the program and gives the order to start the compilation that consists in checking the correct use of the “labels” and the “subprograms” defined in the program. If no errors are found, the program will be stored in the non-volatile memory of the controller.

e.g.: beginning of subprogram

Program made up only of the Main Program.

```

....
....
M02
]
    
```

Program made up of the Main Program and of two subprograms

```

....
....
M02
:10
G51 X+
M99
:20
G51 Y+
M99
]
    
```

**Important**

The prompt is sent as a reply to the controller only when the compilation of the program has been completed: the response time of the controller will vary according as a function of the complexity of the program being compiled.

If an error is found during the compilation, the program will not be stored. The error strings will be sent in the common format for all the errors ('?..') before the prompt.