

USERGUIDE

S900-II

Programming - Level 1

Version 2.2



WARNING - Reliance on this Manual Could Result in Severe Bodily Injury or Death!

This manual is out-of-date and is provided only for its technical information, data and capacities. Portions of this manual detailing procedures or precautions in the operation, inspection, maintenance and repair of the product forming the subject matter of this manual may be inadequate, inaccurate, and/or incomplete and cannot be used, followed, or relied upon. Contact Conair at info@conairgroup.com or 1-800-654-6661 for more current information, warnings, and materials about more recent product manuals containing warnings, information, precautions, and procedures that may be more adequate than those contained in this out-of-date manual.

Logo definitions :



Warning, risk



Sepro robotique inventions



What to do ?



Document evolutions



Handy hints



Example



Innovation or information
concerning a particular
software version

CONTENTS

I – STUDYING AN APPLICATION EXAMPLE	1
I – 1. Description of the robot cycle	1
I – 2. SAP point markers	2
I – 3. Imprecision markers	3
I – 4. Velocity markers	3
I – 5. Time delay markers	3
I – 6. Transcribing the cycle into Sepro S900–II language	4
II – ACCESSING THE PROGRAMMING MODE	9
II – 1. Entering the program	9
II – 1. 1. Accessing the pendant’s editor	9
II – 1. 2. Selecting the instructions	12
II – 1. 3. Entering the subroutines	15
II – 2. The pendant’s editing functions	19
II – 2. 1. Movements in the program	19
II – 2. 2. Deleting an incorrect instruction	19
II – 2. 3. Deleting a step	20
II – 2. 4. Changing a value entered	20
II – 2. 5. Inserting an instruction in a step	20
II – 2. 6. Inserting a step	21
II – 2. 7. Changing the name of a program or subroutine	21
II – 2. 8. Editing the messages	22
III – PROGRAM STRUCTURE	23
III – 1. Main program – PRG00 to 99 –	23
III – 2. Subroutines	23
III – 2. 1. Standard subroutines – SP 01 to SP 40	23
III – 2. 2. Subroutine SP00 : “jump” instruction	24
III – 2. 3. Stacking subroutines – SP 41 to SP 80	25
III – 2. 4. Parallel subroutines – SPP 81 to SPP 99	25
III – 3. Home Return Subroutine – SR –	26
III – 3. 1. Home return subroutines – SR00 to 99 –	26
III – 3. 2. Tool change position subroutine – SR 99 –	29
III – 4. PLC program : 01 to 98	29
III – 5. Advice for cycle time optimalization	30

IV – PROGRAMMING INSTRUCTIONS	32
IV – 1. Predefined actions	32
IV – 1. 1. Bistable pneumatic commands	32
IV – 1. 2. Commands for the injection moulding machine	34
IV – 1. 3. Other predefined commands	36
IV – 2. Instructions	37
IV – 2. 1. Variables	37
IV – 2. 2. Boolean instructions	37
IV – 2. 3. Allocation and operation instructions	38
IV – 2. 4. IF test instruction	40
IV – 2. 5. Time delays : TIME	41
IV – 3. Motorized movement codes	42
IV – 3. 1. Movement code	42
IV – 3. 2. Type of movement	44
IV – 3. 3. Operand	44
IV – 4. The preparatory functions "FUNC" of the numeric axes	45
IV – 4. 1. VEL : Speed axis in% (Maintained)	45
IV – 4. 2. ACC : Axis acceleration in % (Maintained)	45
IV – 4. 3. SLA : Slow approach (Temporary)	46
IV – 4. 4. IMP : Imprecision (Temporary)	47
IV – 4. 5. MASTER : Master movement (Temporary)	48
IV – 4. 6. LINE : Linearity (Temporary)	50
IV – 5. Specific codes	51
IV – 5. 1. SP code as an instruction	51
IV – 5. 2. PLC code as an instruction	51
IV – 5. 3. SR code as an instruction	51
IV – 5. 4. "L" and "R" labels	52
IV – 5. 5. END of PRG, SP..., SR, PLC codes	52
V – SPECIFIC PROGRAMMING	53
V – 1. PLC and parallel subroutines – SPP examples	53
V – 1. 1. Managing a timed belt indexing	53
V – 1. 2. Managing a belt's "step by step" movement	54
V – 1. 3. Maintaining a pulsed input	55
V – 2. System data items	56
V – 2. 1. System bits	56
V – 2. 2. The part counters	58
V – 3. Example of part palletization	59
V – 4. Customized messages	61
V – 4. 1. The comment files	61
V – 4. 2. Customized fault messages	62
V – 5. IMM anticipated restart (option)	64
V – 5. 1. Anticipated restart with programmed delay (Parameter 174 = 2)	65
V – 5. 2. Auto-adaptative anticipated restart (Parameter 174 = 1)	65
V – 6. Changing program automatically	66
V – 7. Example of program with insert placing	67

VI – MEMORY MANAGEMENT	68
VI – 1. The memory explorer functions	68
VI – 2. Memory management in programming mode	70
VII – ANNEX	71
FIGURES	72
INDEX	73

You are advised to read at least the first two chapters of the “S900-II User Manual”.

I – STUDYING AN APPLICATION EXAMPLE

This chapter describes an unloading application from an injection moulding machine (IMM). The example starts with the need analysis and goes as far as entering the program on the S900-II pendant. This example can be used as a basis for all new users of the Sepro S900-II control unit who wish to create programs.

It uses the SAP functions for teaching a cycle. Program structure is described in chapter III – page 23.

The language instructions are described in chapter IV – page 32.

I – 1. Description of the robot cycle

The cycle described in the example is an IMM unloading application with a single part release on a conveyor belt. The cycle is defined by the numbered points.

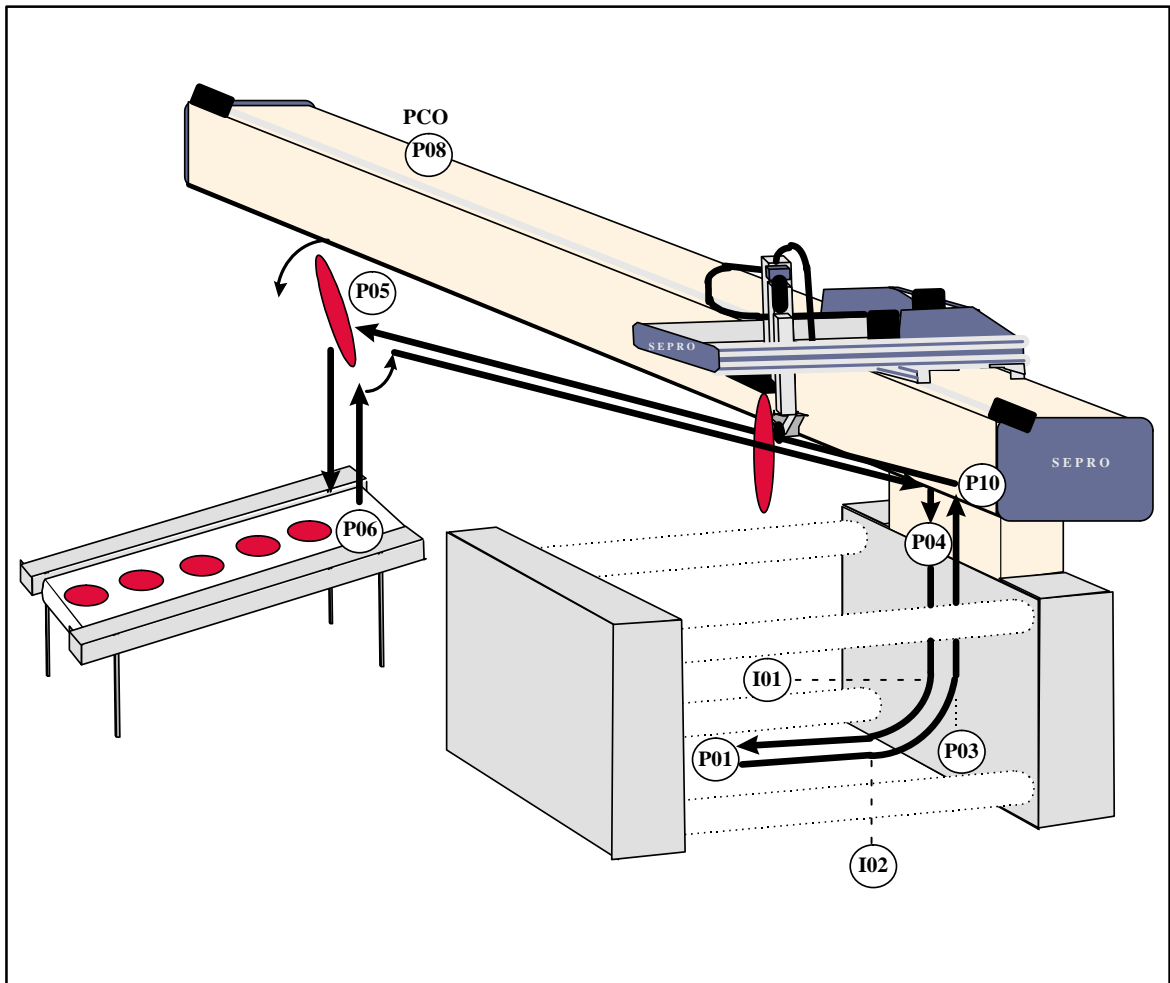


Figure 1 : Cycle movements

<u>THE CYCLE</u>	
The main sequences	The actions and movements
Initial conditions for starting cycle	<ul style="list-style-type: none"> – Release part. – Z complete ascent (P10). – Gripper head vertical.
Start IMM cycle and await opening	<ul style="list-style-type: none"> – X and Y positionned above (P04) the IMM (in the IMM axis, ready to descend). – IMM cycle is started and “ejectors back” validated. – Z approaches mould (P04) staying on the Out of Mould Area cam (ZHM). – Waiting for mould to open.
Part grip in the mould and IMM cycle restart sequence	<ul style="list-style-type: none"> – Z down into the mould } (P01) – Y forward towards the part } (P01) – Ejectors forward and waiting for ejectors completely forward. – Part grip. – Y back to remove part and ejectors back. } (P03) – Z up out of the mould during Y return. } (P03) – IMM cycle restarted during ascent.
Part release on the conveyor belt sequence	<ul style="list-style-type: none"> – X and Y positionned for gripper movement (P05). – Gripper head horizontal. – X and Y positionned above the belt (P06). – Z descent onto the belt (slow speed) (P06). – Part release. – Time delay. – Z complete ascent (slow speed) (P10). – Gripper head vertical.
Belt indexing	<ul style="list-style-type: none"> – Belt indexed for 5 seconds.

I – 2. SAP point markers



To facilitate the modification of position values, Sepro has provided the possibility of assigning markers to axis' positions in the program. This means that to modify the values of positions marked by “point markers”, you do not need to :

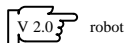
- know the Sepro language,
- access the program using the S900–II editor.

(See User Manual for the modifications).

To define the point markers, you must first analyse the robot's cycle to see where the points should be (See figure 1 : page 1). These points may refer to one or several moving axes.

<u>POINT DEFINITION</u>	
P01 :Grip part in the mould	P06 :Release
P03 :End of return after part grip	P08 :Tool change
P04 :Await mould open	P10 :Arm 1 up *
P05 :Gripper orientation	

Note : The SAP source programs are programs with markers where the axes' position values are declared as needing to be taught, as they can be variable.



I – 3. Imprecision markers

It is possible to give imprecisions markers. These have the same advantages as the point markers, i.e. it is not necessary to :

- know the Sepro language,
- access the program using the S900–II editor.

In this application example, there are only two different imprecisions :

- Y forward anticipation during Z descent into the mould,
- Z ascent anticipation during Y back.

I01 :Anticipation of Y advance

I02 :Anticipation of Z ascent

I – 4. Velocity markers

It is possible to assign markers to velocities. These offer the same advantages as the point markers, i.e. it is not necessary to :

- know the Sepro language,
- access the program using the S900–II editor.

In this application example, there are only two different speeds which are for all the axes :

- slow speed for Z descent to belt (then ascent),
- the rest of the time : max. speed for all the axes.

V01 :Z speed for descent in mould

V07 :Z descent speed for release

I – 5. Time delay markers

As for the velocities, it is possible to assign markers to time delays.

Two time delays are used in this application example :

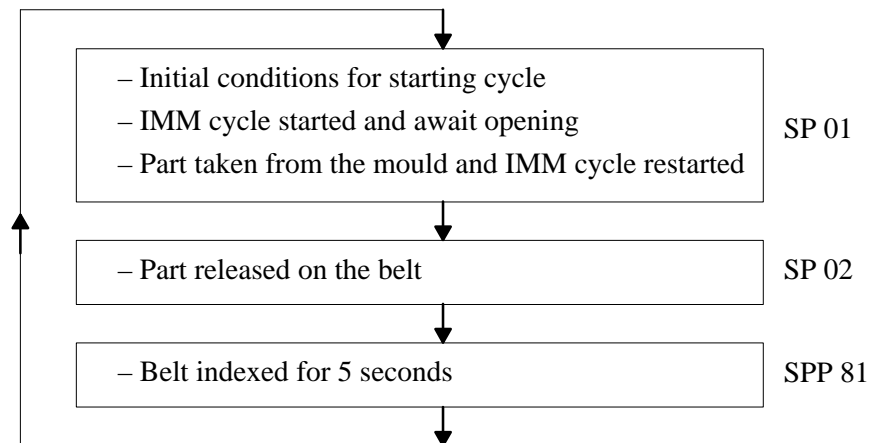
T01 :Time delay after part release

T02 :Conveyor command time

I – 6. Transcribing the cycle into Sepro S900–II language▶ The main program PRG

The advantages of separating the cycle into subroutines as shown in the following organigram are that :

- the program is easier to read,
- the program is easier to change,
- the IMM is immobile for a shorter length of time.



The main program is made up of call-ups of different subroutines.

```

SAP source program number : 99
*[APPLICATION EXAMPLE      ]
*
STEP  000
  PLC      00
  SP  01   L00      PART GRIP IN MOULD
STEP  001
  SP  02   L00      PART RELEASE ON BELT
STEP  002
  SP  81   L00      BELT INDEXING DELAYED
STEP  003
  END
  
```

```
SUBROUTINE ->STANDARD 01
*[PART GRIP IN MOULD      ]
*
STEP 000
  Release part 1
  Z.ABS_L 00100.0 P 10 Arm 1 up
STEP 001
  Gripper vertical
STEP 002
  X.ABS_L 00170.0 P 04 Await mould open
  Y.ABS_L TEACH. P 04 Await mould open
STEP 003
  Z.ABS_L TEACH. P 04 Await mould open
  Await end of machine cycle 1
  Ejectors 1 in
  Stop validation ejectors 1 out
STEP 004
  Z.ABS_L TEACH. P 01 Grip part in the mould
  IMP.Z 005.0 I 01 Anticipation of Y advance
STEP 005
  Y.ABS_L TEACH. P 01 Grip part in the mould
STEP 006
  Stop validation ejectors 1 in
  Ejectors 1 out controlled
  Grip part 1
STEP 007
  Ejectors 1 in
  Stop validation ejectors 1 out
  Y.ABS_L TEACH. P 03 End of return after part grip
  IMP.Y 005.0 I 02 Anticipation of Z ascent
STEP 008
  Z.ABS_L 00110.0 P 10 Arm 1 Up
  Validation machine cycle 1
STEP 009
  END
```

```
SUBROUTINE ->STANDARD 02
*[PART RELEASE ON BELT          ]
*
STEP 000
  X.ABS_L TEACH. P 05 Gripper orientation
  Y.ABS_L TEACH. P 05 Gripper orientation
STEP 001
  Gripper horizontal
STEP 002
  X.ABS_L TEACH. P 06 Release
  Y.ABS_L TEACH. P 06 Release
STEP 003
  Z.ABS_L TEACH. P 06 Release
  VEL.Z   020   V 07 Z descent speed for release
STEP 004
  Release part 1
STEP 005
  TIME    015   T 01 Time delay after part release
STEP 006
  Z.ABS_L 00100.0 P 10 Arm 1 up
STEP 007
  VEL.Z   100   V 01 Z speed for descent in mould
  Gripper vertical
STEP 008
  END
*
```

```
SUBROUTINE ->PARALLEL 81
*[BELT INDEXING DELAYED        ]
*
STEP 000
  OUT     099   BELT INDEXING OUTPUT
STEP 001
  TIME    050   T 02 Conveyor command time
STEP 002
  END
*
```

► Home return subroutine : SR00

This subroutine is absolutely necessary for the program to work. Its function is described in chapter III – 3. page 26.

For this application example, we can distinguish between two different types of freeing sequences :

1. The robot is in the mould, the following is therefore necessary :

- a part release,
- Y back (P04),
- a Z ascent (P10) (complete and fast).

2. The robot is in the release position (just above the belt), the following is therefore necessary :

- a part release,
- a Z ascent (P10) (complete and fast).

Note : In the home returns, it is not necessary to carry out the movements that will be repeated at the beginning of the program, such as the rotations or the X movement. You just need to move the robot arm (Z axis) to a safe position (often, arm up).

When comparing the two types of freeing sequences in this example, only one difference can be noted : the Y return is only necessary for freeing from the mould.

Therefore, the solution adopted that will be compatible with all the positions is :

- release part,
- if the robot is in the machine axis (AM) then Y back (P03),
- Z ascent (P10) (complete and fast).

SR00 is written in Sepro S900–II language.

```
HOME-RETURN SUBROUTINE  00
*[HOME RETURN           ]
*
*
STEP  000
  Release part 1
STEP  001
  IF IN  017      ROBOT ON MACHINE AXIS CAM
  Y.ABS_L  TEACH. P 03 End of return after part grip
STEP  002
  Z.ABS_L  00100.0 P 10 Arm 1 up
  VEL.Z    100      V 01 Z speed for descent in mould
STEP  003
  Ejectors 1 in
  Ejectors 1 out
  SP  81  L00      BELT INDEXING DELAYED
STEP  004
  END
*
```

► Tool change position subroutine : SR99

Its function is described in chapter III – 3. 2. page 29.

Reminder : when a tool change position is requested , the robot first carries out a home return



Therefore, the contents of SR 99 are :

X movement (P08) towards tool change position :

```
HOME-RETURN SUBROUTINE 99
*[TOOL CHANGING POSITION      ]
*
*
STEP 000
  X.ABS_L TEACH. P 08 Tool change
STEP 001
  END
*
```

II – ACCESSING THE PROGRAMMING MODE

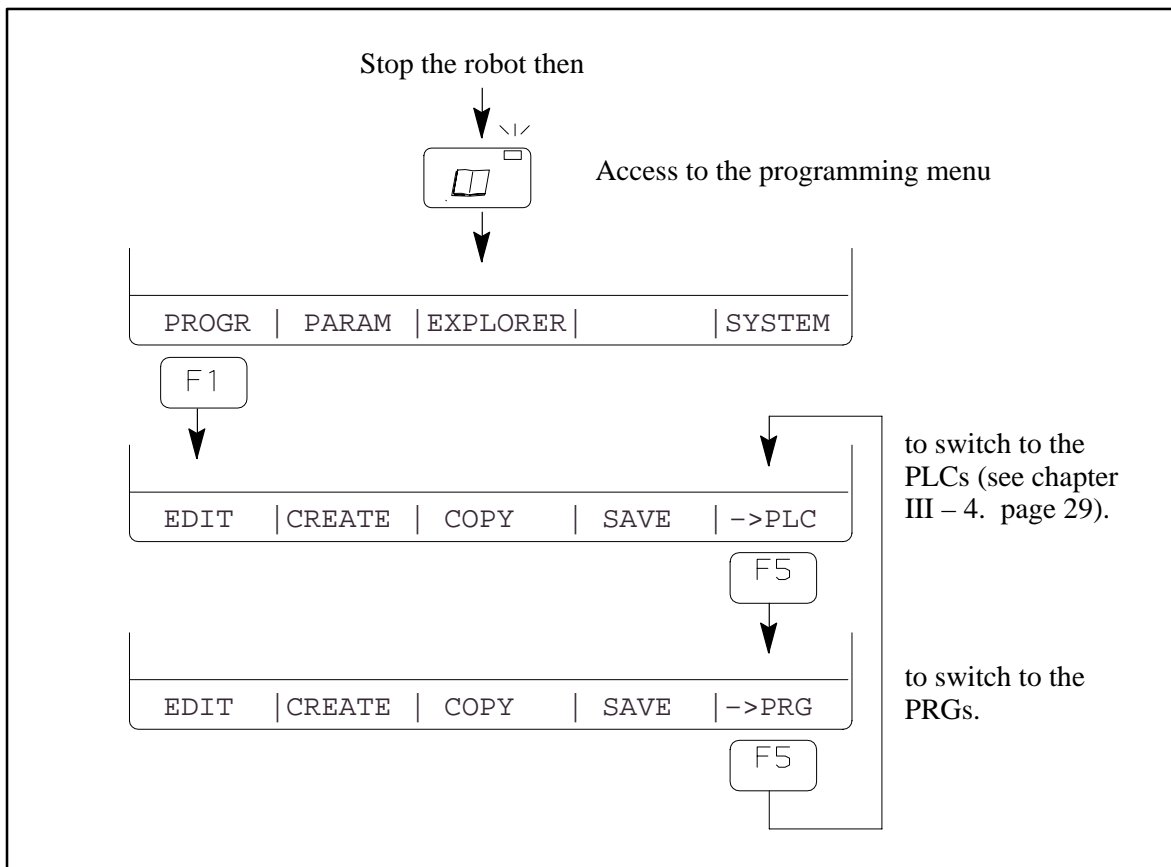
II – 1. Entering the program

There are two ways of entering the program :

- ▶ directly on the robot's pendant (as shown in the example in chapter I – page 1),
- ▶ on a PC equipped with the AS900–II software (see the AS900–II editor for PC documentation).

II – 1. 1. Accessing the pendant's editor

Creating a program with the S900–II robot editor is possible if the robot is not in production. This is shown in the example on page 1 . Details on the editing functions are given in chapter II – 2. page 19.



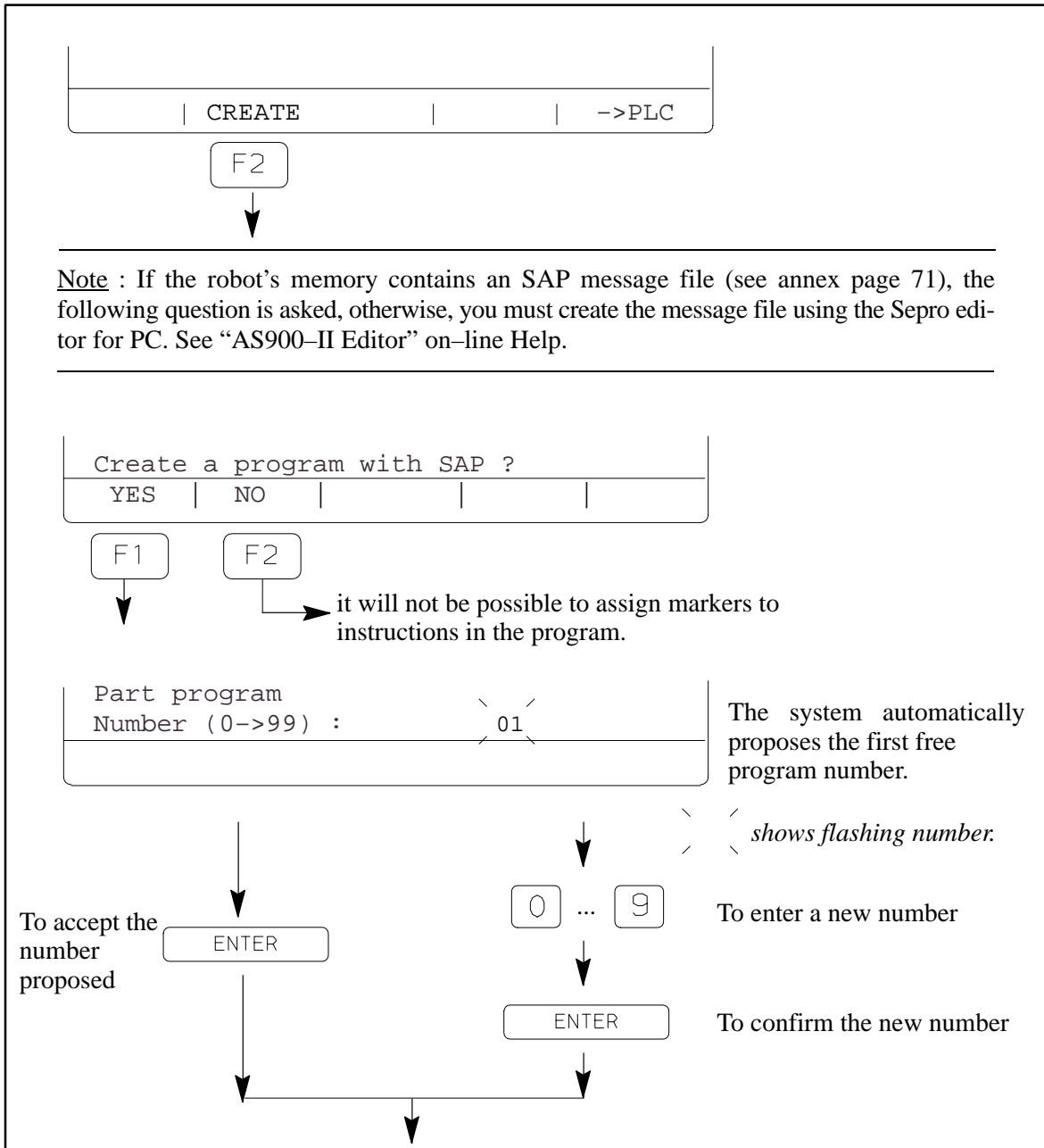
Note : If the robot's memory does not contain the type of program selected, the following menu

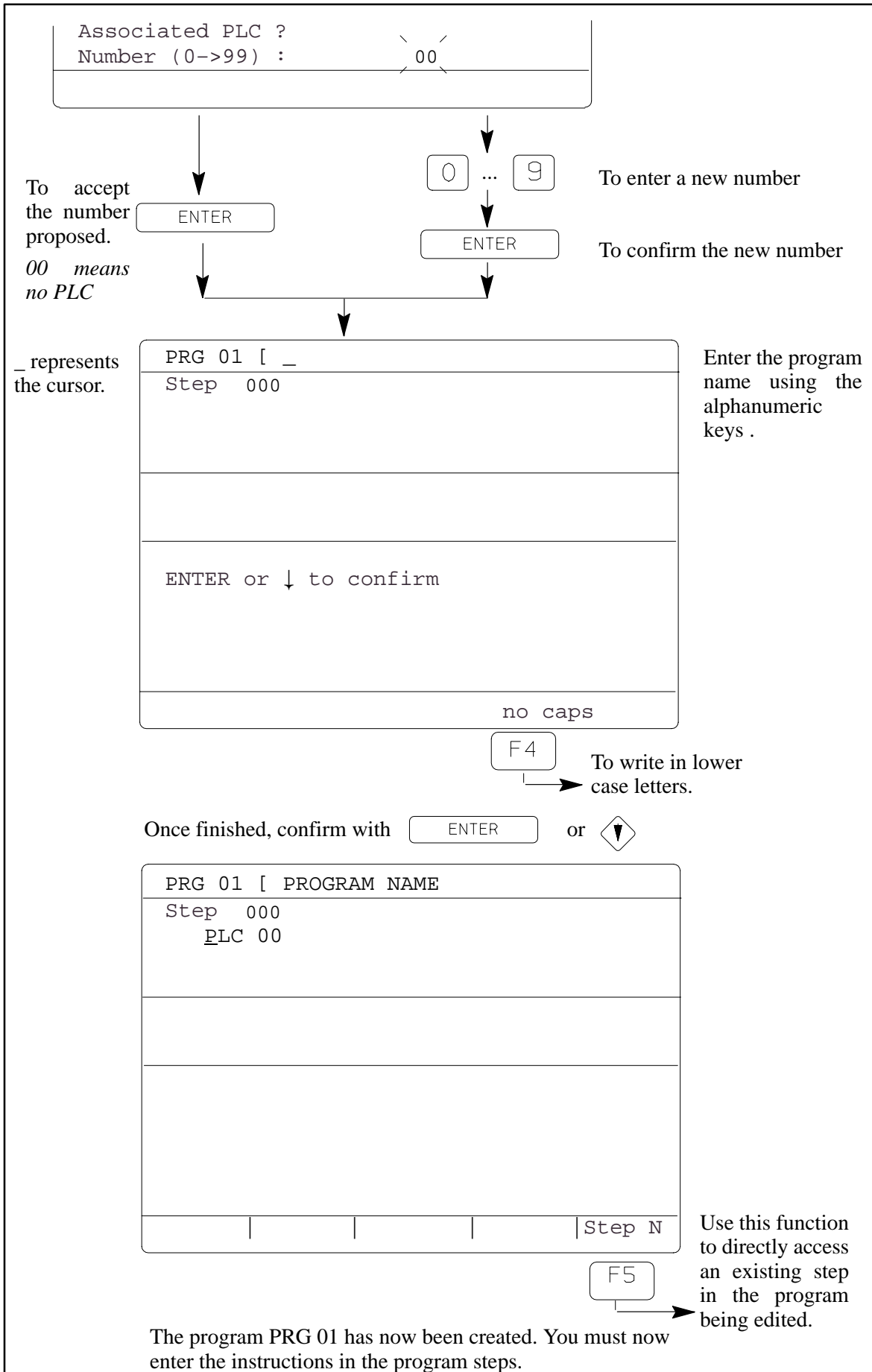


To create a new part program, you must be in the part program window (PRG) with the PLC function

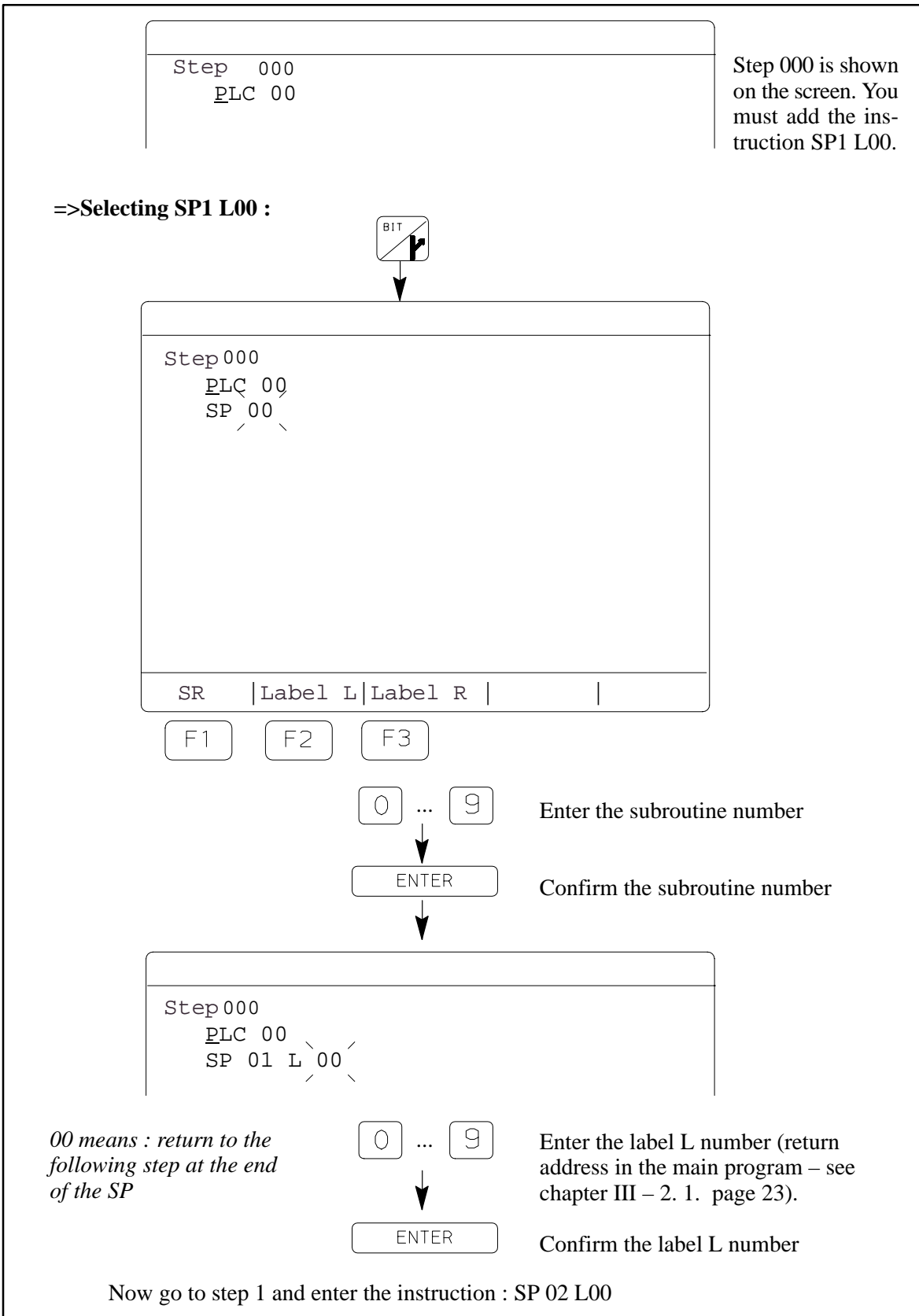
above **F5** in the menu.

Whatever the menu, the new program is entered by selecting :

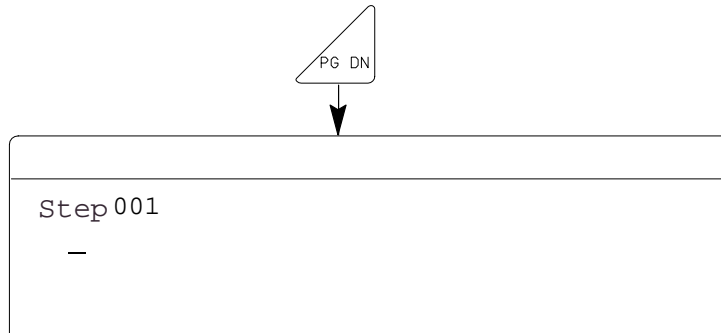




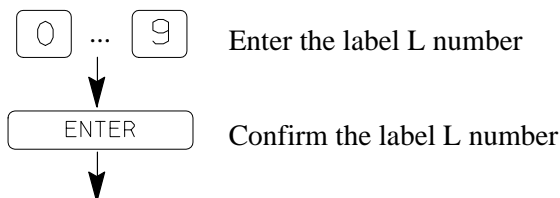
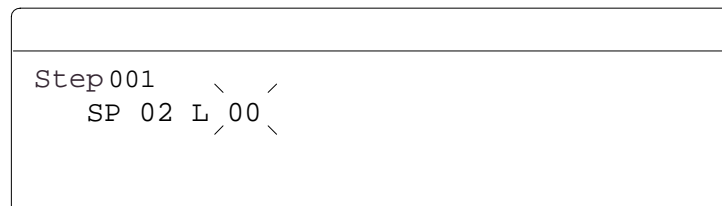
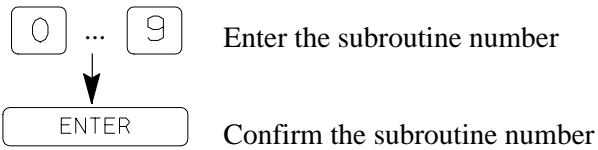
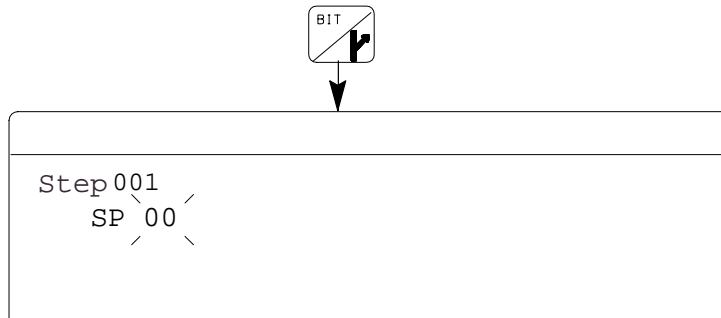
II – 1. 2. Selecting the instructions



=> Moving to step 1 :



=>Selecting instruction SP2 L00 :



Now go to step 2 and enter the instruction : SP 81 L00

=> **Moving on to the following step :**



Step002 —

Enter instruction SP81 L00 in the same way as the preceding SPs.

Step002 SP 81 L00

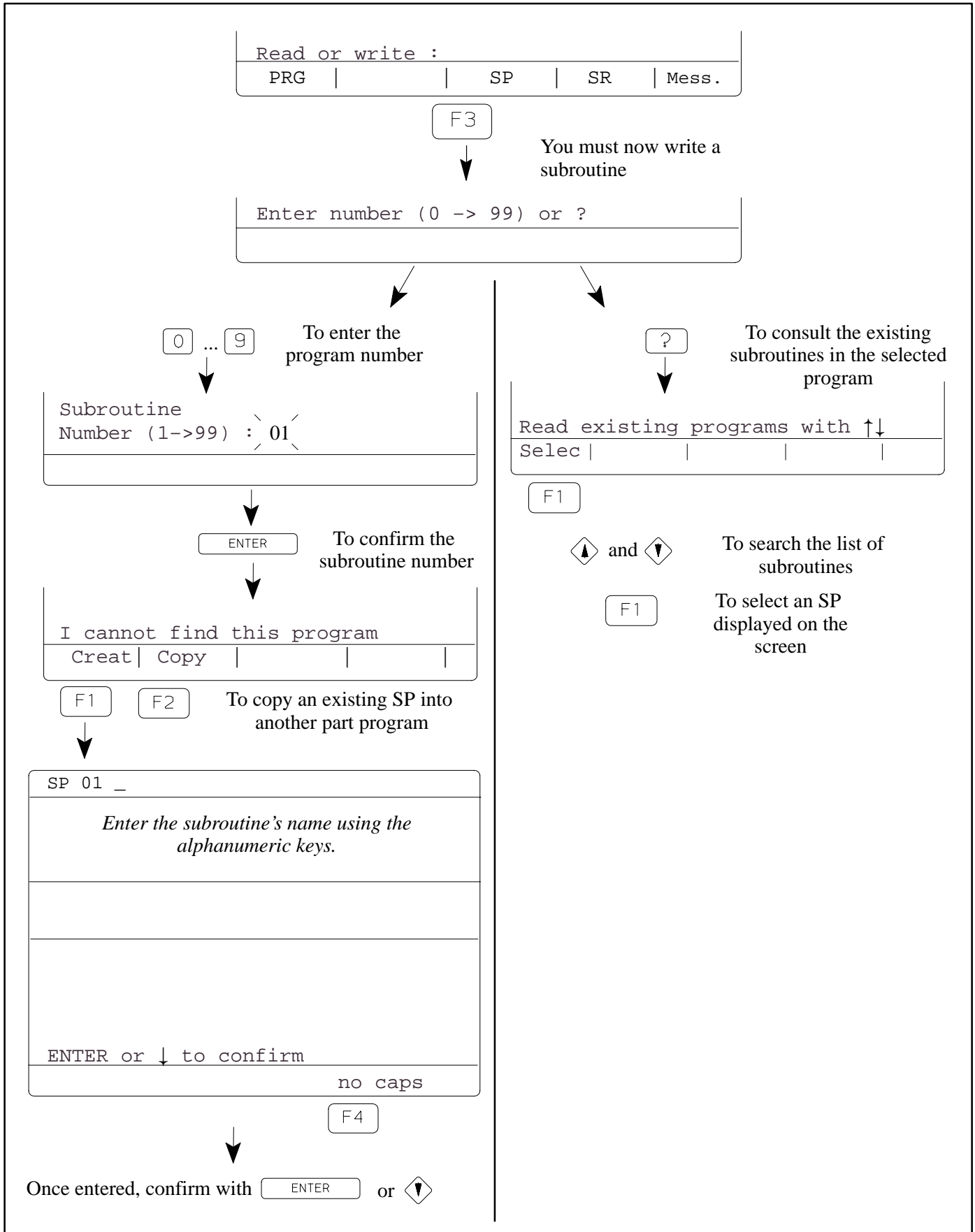
The main program has now been entered. You must quit the program to write the subroutines.

=> **Quitting the main program :**

Press

Note : The “END” instruction is automatically inserted once has been pressed. See chapter IV – 5. 5. page 52.

II - 1. 3. Entering the subroutines



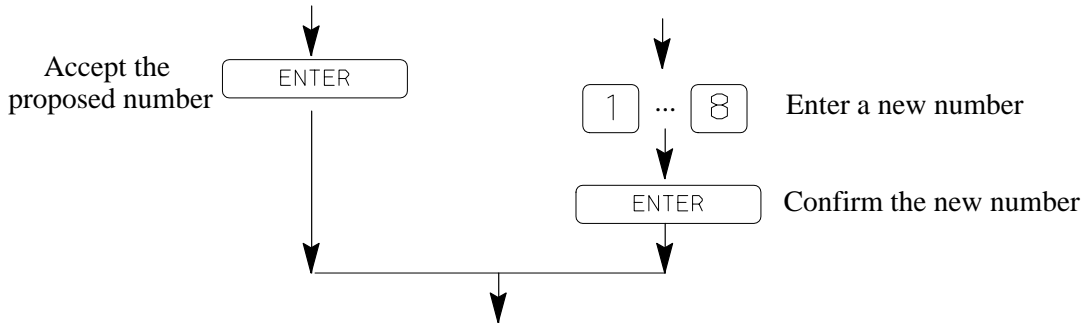
=> Selecting "release part 1" :



```

SP01 [ PART GRIP IN MOULD ]
Step000
Release part` 1 `
      `  `
      `  `

Number (1->8) : 1
    
```



=> Selecting Z.ABS.L 100.0 :



or



to program the Z vertical axis

```

SP 01 [ PART GRIP IN MOULD ]
Step000
Grip part 1
Z.ABS.L 00000.0
      `  `
      `  `

STK | REL | CTL | FREE | .. / ..
    
```

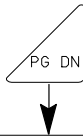
By default, the movement selected is an ABSolute movement. Enter the value 100.0 using the alphanumeric keys then confirm with

```
SAP Point (0->40 0=without) : 00
```

You must now select the SAP point marker P10 using the numeric keys then confirm with

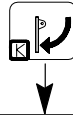
Now go to step 1 and enter the instruction : “Gripper vertical “.

=> Moving to Step 1 :



```
SP01 [PART GRIP IN MOULD ]  
Step001  
—
```

=> Selecting the vertical gripper head :

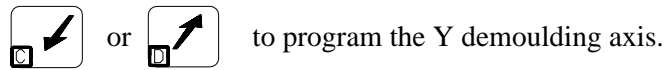
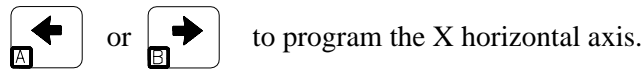


```
SP01 [PART GRIP IN MOULD ]  
Step 001  
Gripper vertical
```

=> Moving to Step 2 :



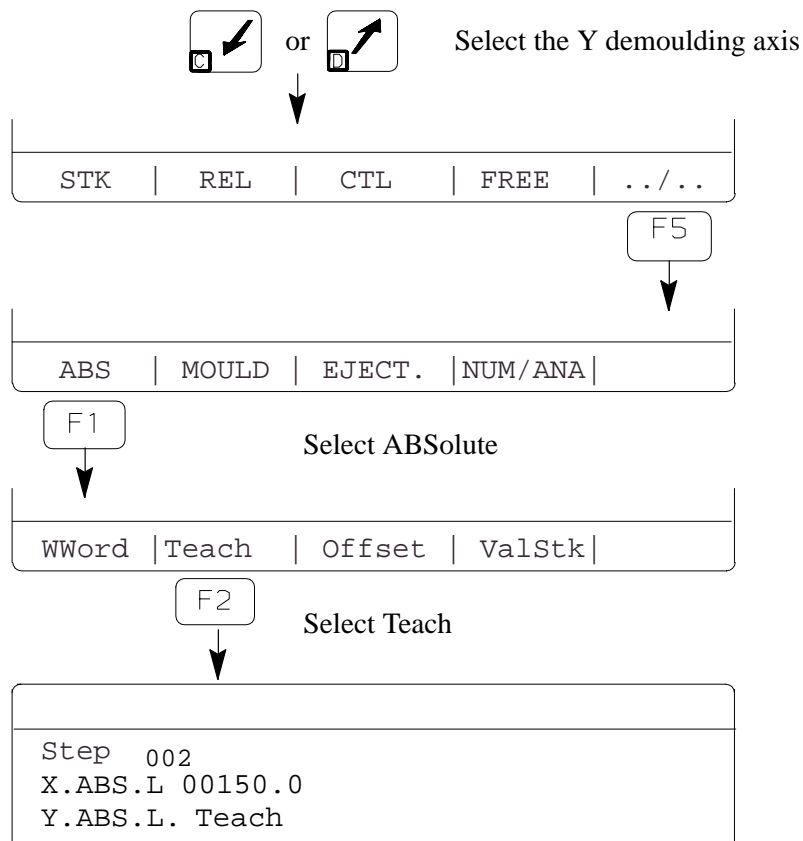
=> Entering the instructions X.ABS.L 150.0 and Y.ABS.L Teach :



X.ABS.L 150.0 are obtained in the same way as Z.ABS.L 100.0.

Entering a fixed value corresponds to the robot's stopping position. The TEACH instruction means that the position will be taught during the first execution.

To write the instruction Y.ABS.L Teach, you must :



Continue entering the instructions for SP1, referring if necessary to chapter IV – Page 32 and to the examples on pages 3 to 7. At the end of subroutine 1 (SP1), quit the subroutine (SP) by pressing

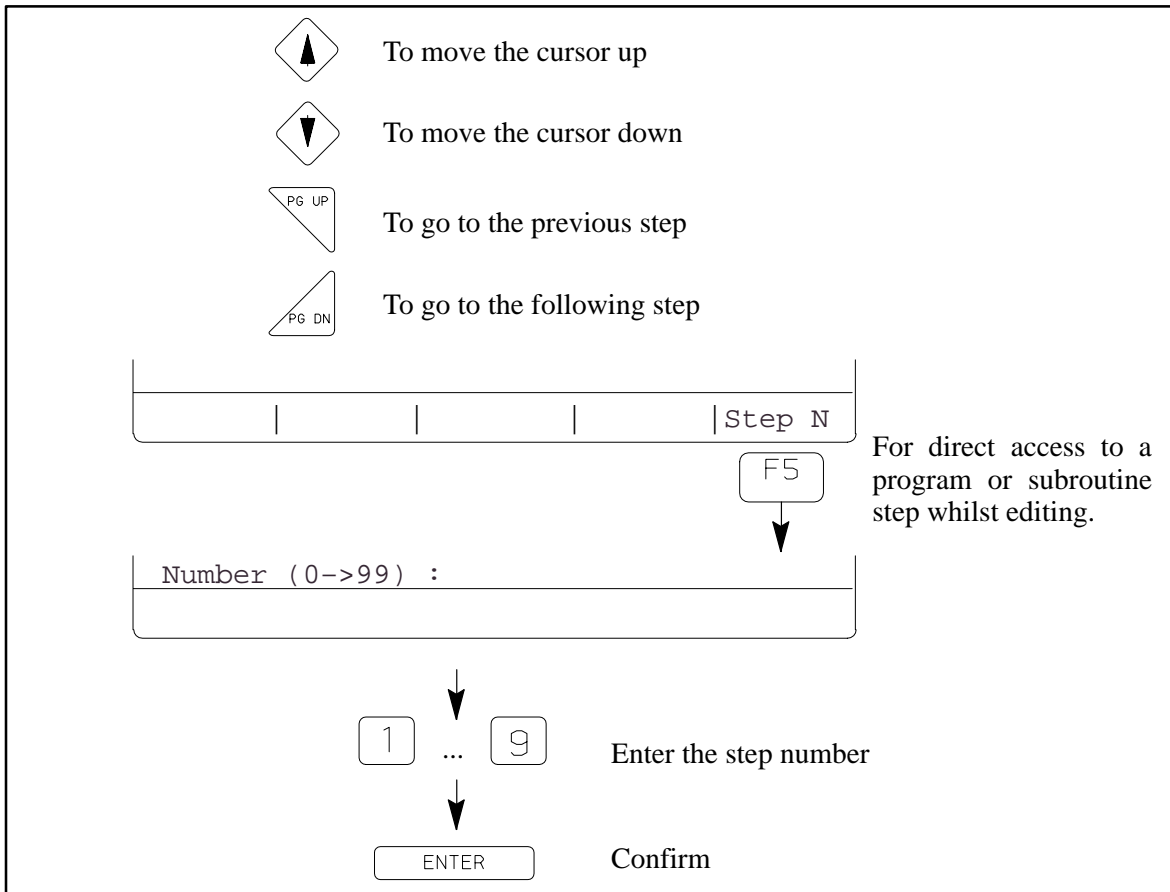
. Then create subroutine 2 (SP2) in the same way as subroutine 1 (SP1).

Repeat the same operations to create the parallel subroutine (SPP 81) and the home return subroutines (SR00 and SR99).

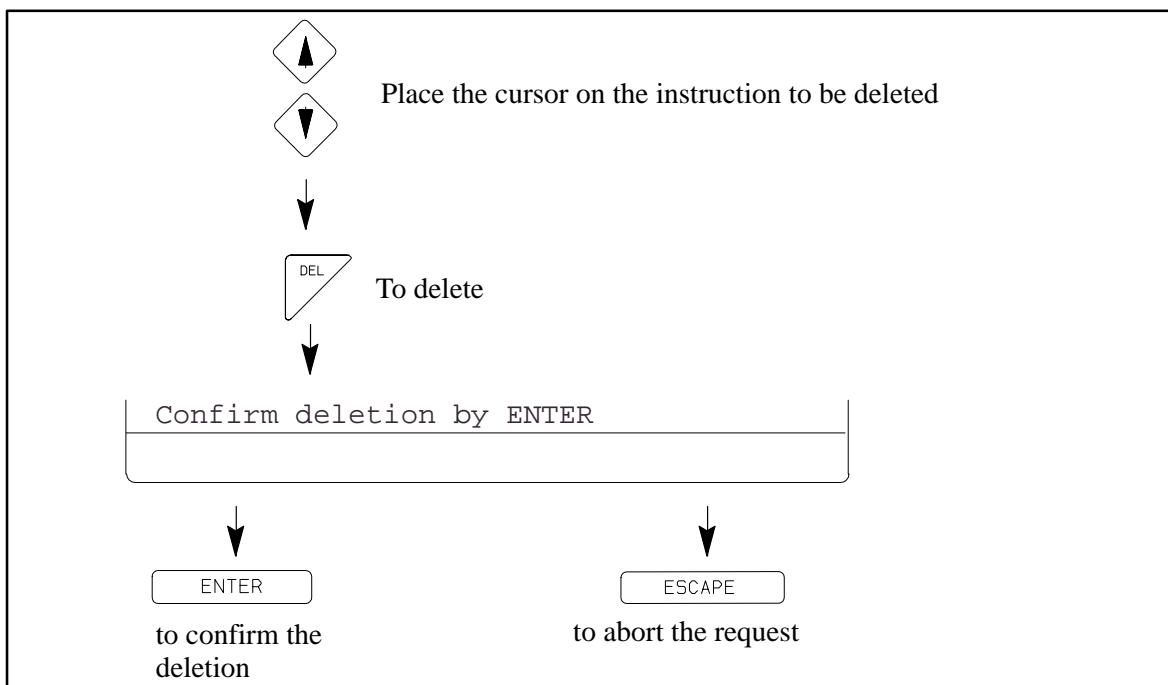
The program has now been entered. Carry out the tests on the program, referring to the S900–II User Manual.

II – 2. The pendant's editing functions

II – 2. 1. Movements in the program



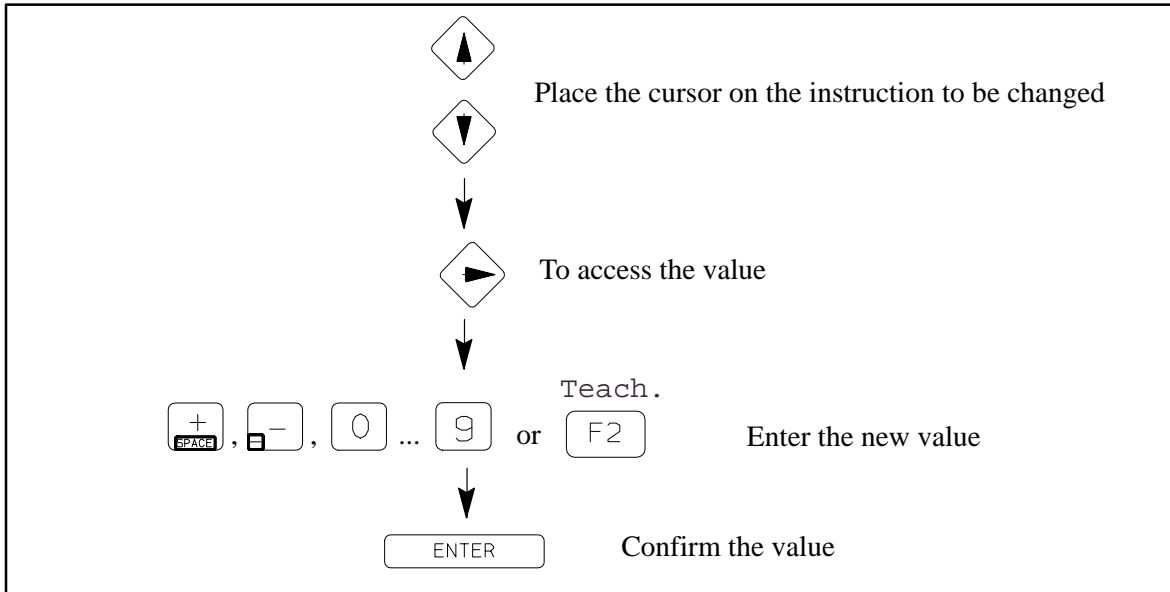
II – 2. 2. Deleting an incorrect instruction



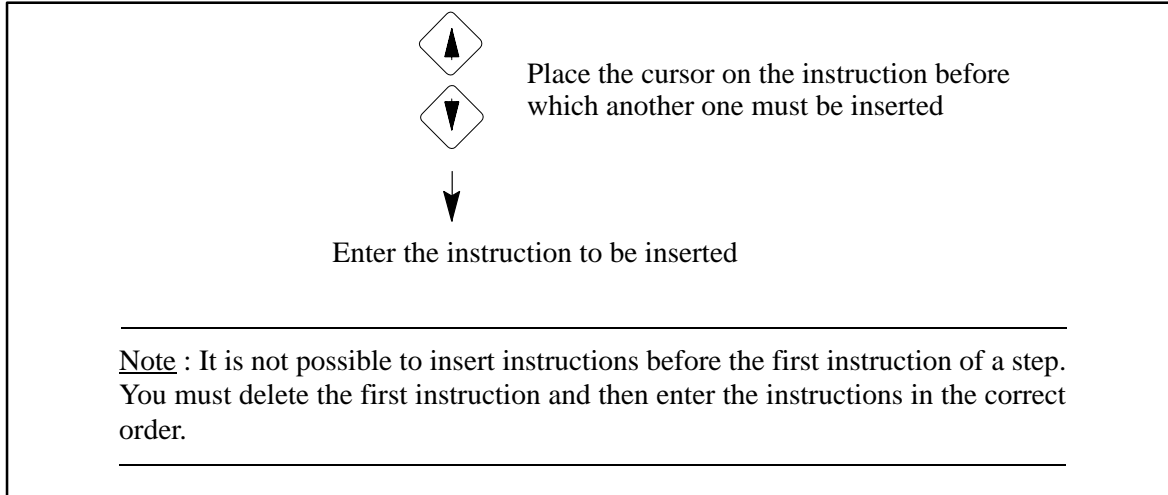
II – 2. 3. Deleting a step

To delete a step, you must delete each instruction contained in the step.

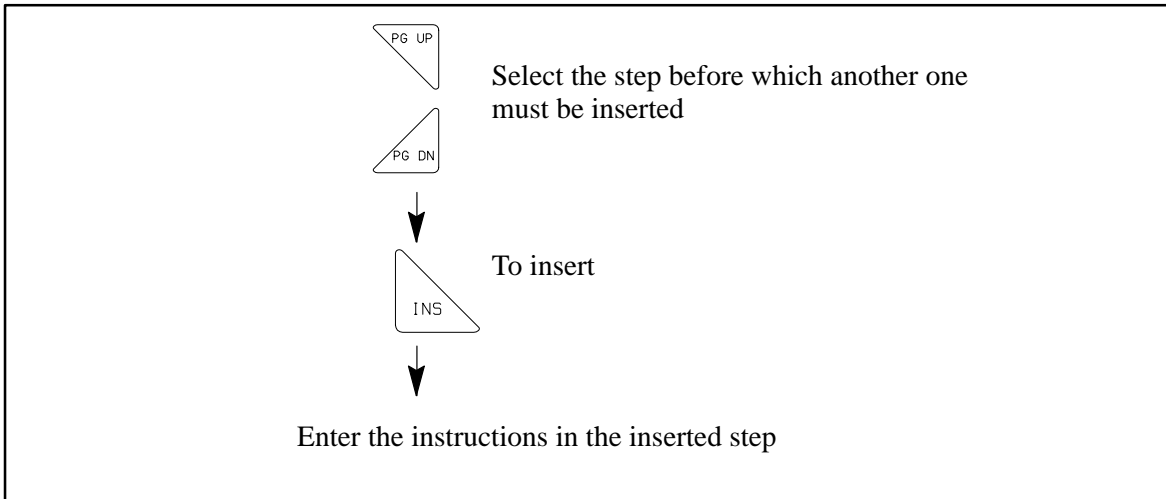
II – 2. 4. Changing a value entered



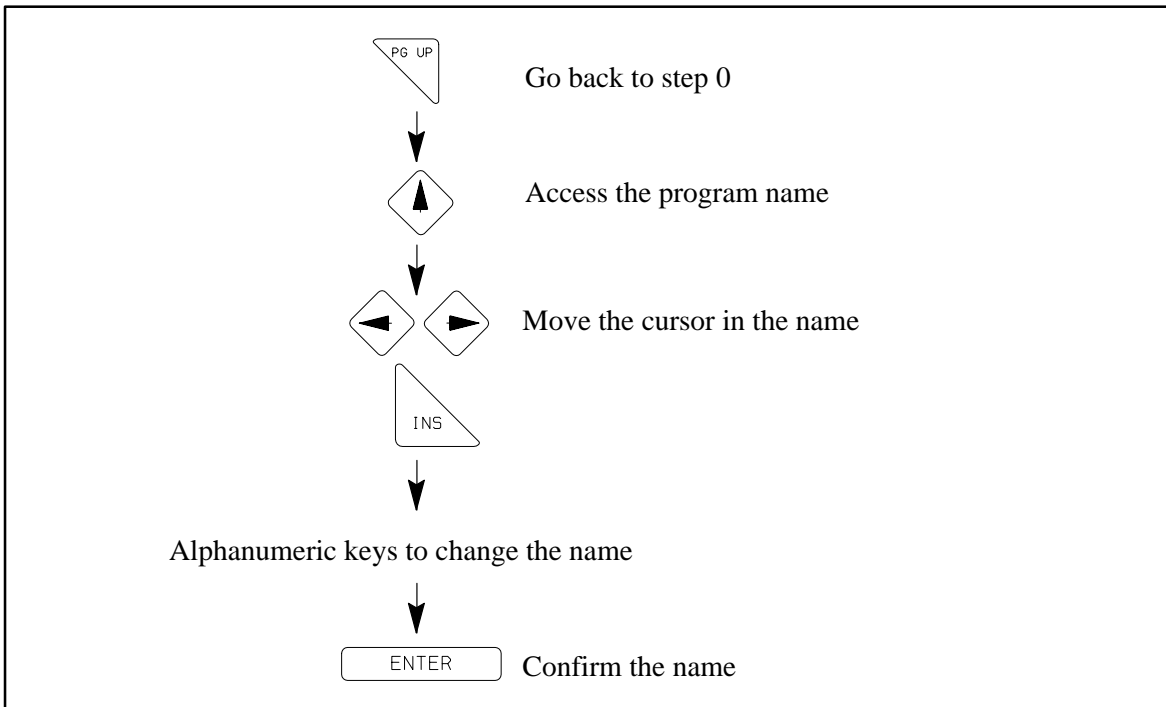
II – 2. 5. Inserting an instruction in a step




II – 2. 6. Inserting a step



II – 2. 7. Changing the name of a program or subroutine

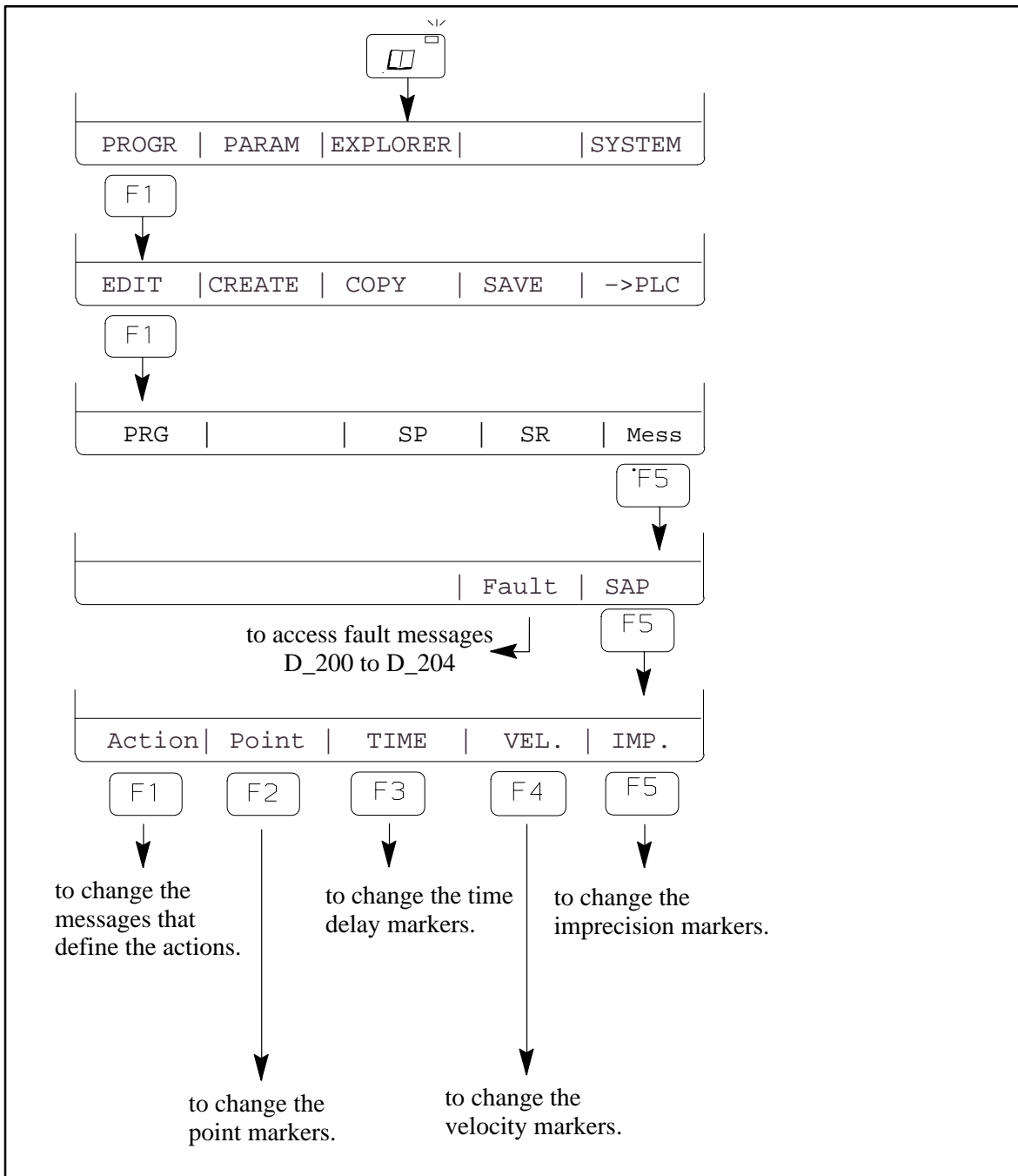


Delete characters already entered using the  key.

II – 2. 8. Editing the messages

There are two types of messages :

- ▶ the SAP messages : that must be created with the Sepro PC Editor and that can be changed on the robot’s pendant (see below).
- ▶ the fault messages : that can be created and changed with the Sepro PC Editor or directly on the robot’s pendant (see page 62).



III – PROGRAM STRUCTURE

III – 1. Main program – PRG00 to 99 –

One hundred 1,000–step main programs (N° 0 to N° 99) can be run and stored in memory. Simultaneous storage only depends on available memory capacity and the size of the routines.

Programs can be “named” (maximum 30 characters) in order to identify them with the product being handled:



”Cover D120 Mould 96032”

It is possible to change this name and display it during program search procedures. See chapter II – 2. 7. page 21.

Structure is sequential, i.e. a step is not considered finished, and therefore the subsequent step cannot be run, until all the instructions it contains have been executed.

III – 2. Subroutines

Subroutines are a series of instructions, structured in a sequential manner.

III – 2. 1. Standard subroutines – SP 01 to SP 40

These are a series of instructions which are grouped together in independant structures and run sequentially.

Like the main programs, they can be “named” (maximum 30 characters) in order to identify them with their function in the program:



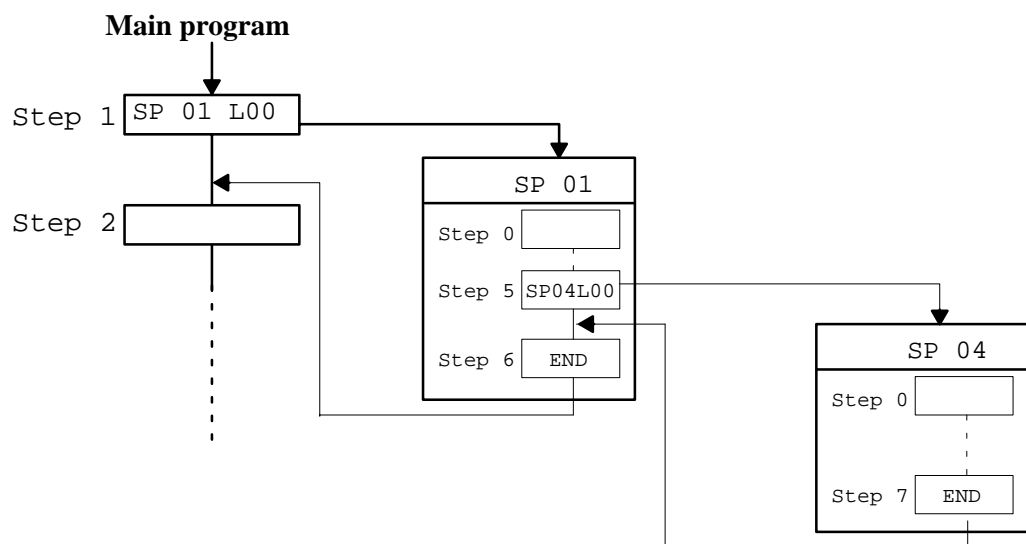
”SP 01 : Grip part in mould”

The address to which they return when the subroutine has been executed (Label L) is declared in the main program. The subroutine call–up is followed by a Label L number. If you put Label 00, you will return to the step following the one in which the subroutine was called, once the subroutine has finished.

A subroutine may call another subroutine, up to a maximum of 3.

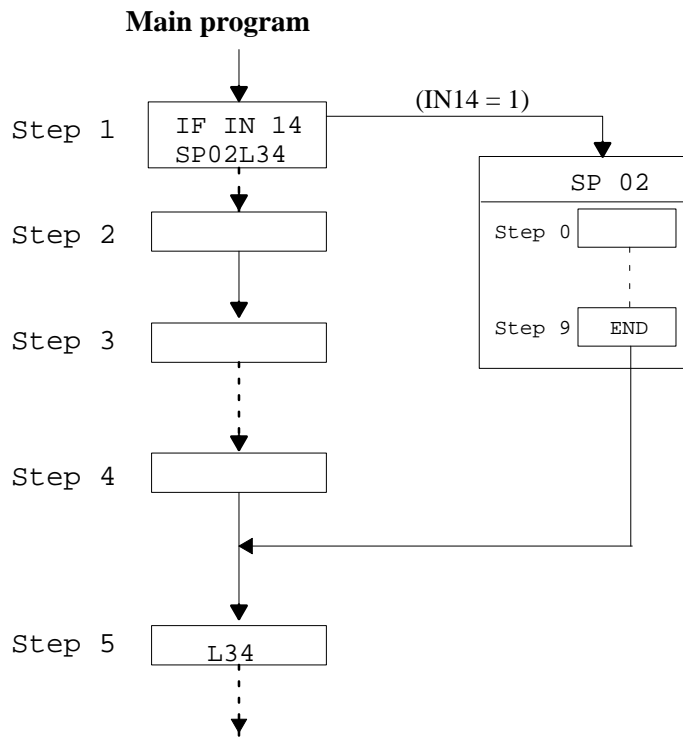


Example No. 1 : Non–conditional execution and return to step following the call step.





Example No. 2 : Conditional execution and return to main program step other than the step following the call step.



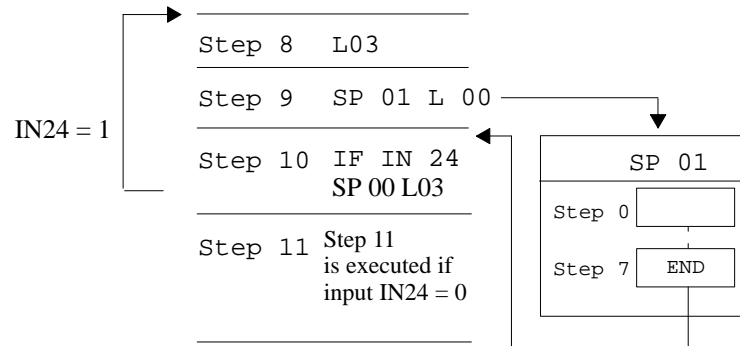
Note that this structure allows you to use a same subroutine with different return addresses : the return address is an integral part of the call instruction. If the return address is other than L00, **it always goes back to a main program step irrespective of the starting point (PRG or SP).**

Conditional execution involves use of the “IF” instruction which is described in the section on subroutine instructions in chapter IV – 2. 4. page 40.

III – 2. 2. Subroutine SP00 : “jump” instruction

Subroutine SP00 is not a standard subroutine. SP00 is a “jump” instruction.

Some programs require jump instructions to return to, or jump to, a given STEP.



Note : Label L00 is not valid : SP00 L00 is not allowed.

III – 2. 3. Stacking subroutines – SP 41 to SP 80

SP 41 to SP 60 : These subroutines are used to simplify the definition of pallets where a stack / column organization already exists. An example is given in chapter V – 3. page 59.

SP 61 to SP 80 : These subroutines are used to describe an irregular stacking of parts which is repeated over several layers or several times in a cycle.

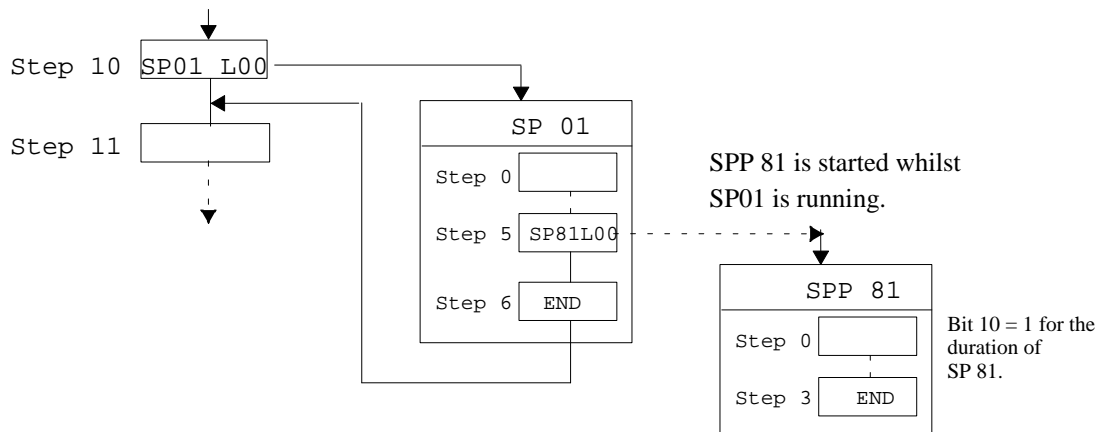
See Programming manual S900–II Level 2.

III – 2. 4. Parallel subroutines – SPP 81 to SPP 99

These subroutines are also groups of instructions, structured in a sequential manner.

They are executed in parallel to the normal running of the cycle.

There is no return address for the Parallel subroutines.



The system bit number 10 is set to 1 when the SPP is started and goes back to 0 at the end of the SPP.

A few restrictions :

- ▶ Only one Parallel Subroutine can be active at a certain time. The calling-up of a second one before the first one is finished, triggers the fault `D_73: PARALLEL SP ALREADY RUNNING`.
- ▶ In an SPP, all the instructions of the main program can be used, apart from :
 - the MASTER preparatory function and consequently all triggered and control (CTL) movements,
 - the special instructions `Await machine cycle` and `Await validation PRG change`.
 - Calling-up of other SPs or SRs.

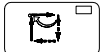
Moreover, for robots using PIP parameters, the following instructions are taken into account during machine dialogue in the part program only. They are ignored in the SPP :

- ▶ `Machine cycle validation`
- ▶ `part grips in the IMM`.


III – 3. Home Return Subroutine – SR –

III – 3. 1. Home return subroutines – SR00 to 99 –

If the robot cycle is interrupted, it will not always be possible to continue the cycle execution from where it was interrupted. The operator can free the robot from this blocked situation by carrying out

a home return .


”Home Return” is a subroutine that takes the robot to a safe position before safely launching the

”Automatic”  cycle.

The Home return Subroutines (SR00 and SR99) enable different robot freeing sequences.

There are two ways of executing a home return :

► when the operator requests it : in Step by Step  mode then  then  held down.

► when there is a part grip fault in the mould : in automatic mode .

Subroutines SR00 to SR99 are valid for each main program.

Like the main programs, they can be ”named” (maximum 30 characters) in order to identify them with their function in the program :



”SR 35 : CLEAR SCALE”

The Return subroutine executed is SR00 if nothing particular has been specified in the program. If you wish to use a Return subroutine (SR) other than SR00, this must be specified in each program or subroutine step where you wish to use it.

Contents of a Return Subroutine :

In a Return Subroutine (SR), the robot freeing sequence must take into account the difficult positions in the cycle (gripper head in the mould or on a peripheral).

It describes the successive movements to be carried out to free the robot from each difficult position.

Note that these structures enable you to use the same subroutine with different return addresses ; they always lead back to a step in the main program, irrespective of the starting point (PRG or SP).

The SRxx instruction, placed in the main program or subroutine steps, enables you to execute SR number xx instead of SR00 when you ask for a home return. See figure 2 : page 28 and the special cases TIME page 41 and MASTER page 48



The execution of Home Return subroutines is sequential and they cannot be restarted until they have been completed.

If the Home Return is interrupted by a change to Adjust mode and if it is forced (parameter 15=1), then the Home Return will start again at Step 0.

Difference between the Simple and Total Home Return :

Home Return [Simple]

When you select a home return [Simple] (selected by default), the robot frees itself and starts its cycle again either from the beginning (step 00 of PRG), or from a specific point in the cycle if this has been programmed. See figure 2 : page 28.

Home Return [Total]

When you select a home return [Total] (key), the robot frees itself and systematically starts again at the beginning of the cycle (step 00 of PRG). The stacking subroutine counters are set to zero, which means that you must clear the pallets in the case of a total home return. For this, the system bit number 9 is set to 1 during the execution of the total home return. See figure 2 : page 28.

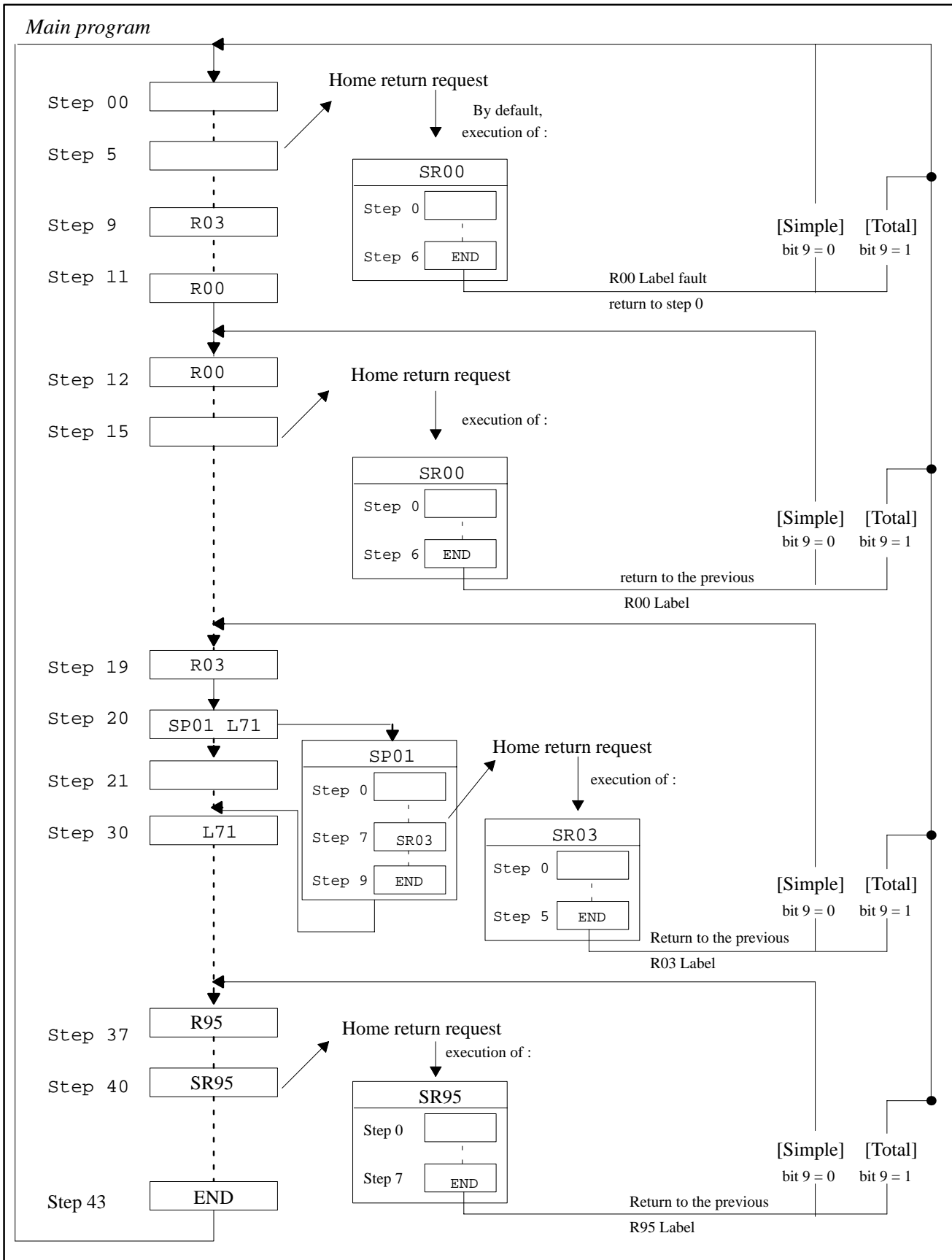





Figure 2 : Return subroutine branches

III – 3. 2. Tool change position subroutine – SR 99 –


The usual abbreviation for a Tool Change Position is "PCO". This procedure is used to free the robot to change the gripping tool or to transfer the robot to a non-operational area in order to operate the host machine without the robot. This position usually corresponds to the robot location where the end of the PCO stroke is on the PCO cam.

Executing this procedure in Step by Step mode , after having pressed  then , initiates the following sequences :

- ▶ the Home return subroutine programmed at the step in which the robot stopped : SR00 by default,

then :

- ▶ the Tool Change Position subroutine, SR99.

Note : these two sequences follow one after the other without having to release the  key.

III – 4. PLC program : 01 to 98

The PLC is a programmable logic controller integrated into the S900–II numeric control system. Therefore, they share their inputs and outputs.

They enable you to manage peripheral units equipped with "ON / OFF" inputs and outputs.

The PLC's scan time is 100 ms which enables you to monitor one or several peripheral units whose cycle, complete or partial, is asynchronous to the robot's cycle.



. Monitoring a spacer stacking manipulator.



. Controlling a pile of boxes filled by an operator at any point in the cycle.

Each main program can call up a PLC. The same PLC can be used in several main programs.

PLC examples are presented in chapter V – 1. page 53.

Reminder : PLC 00 does not exist. A main program that starts with the instruction PLC 00 does not use a PLC.

PLC 99 is reserved for monitoring special safety conditions linked to the robot's environment.

The monitoring PLC is described in the Programming manual Level 2.

III – 5. Advice for cycle time optimization

The IMM cycle time is often greater than the robot's one. Even so, it is always interesting to reduce it, especially the IMM immobilization time.

The IMM immobilization time is the length of time that separates :

- the mould access authorization that the IMM gives to the robot : Mould Open (MO) or Partial Opening Reached (OPA) signal
from
- the Machine Cycle Validation (VCM) that the robot gives to the IMM to authorize the fabrication of a new part.

To reduce the IMM immobilization time, you can use a certain number of instructions and options offered by Sepro, and several rules must be respected to guarantee the equipment safety and the reliability of the cycle.

- ▶ The cycle time reduction **MUST NEVER** increase the number of incidents. In this case, the time lost restarting after an incident cancels the few tenths of a second gained during optimization.
- ▶ The paths obtained after optimization must not contain a risk of collision if, for one reason or another, a part of the movement slows down.

For example, when you switch to Step by Step mode :



The robot arm must be stopped as close as possible to the mould when waiting for the latter to open. The IMM cycle must be restarted as soon as possible. To ensure this, the Out of Mould Area (ZHM) cam is adjusted as correctly as possible : the lowest position of the robot's arm that allows the mould to open and close without colliding with the robot's gripper head.

The Sepro programming instructions used for the cycle time optimization are imprecise (IMP) and the master movement (MASTER). Their use is described in chapters IV – 4. 4. and IV – 4. 5. page 47.

Using these two tools enables you :

- to round the paths as the movement follow each other
- to mask the part demoulding movements (ejectors and/or core pullers) by doing them during the robot's movements.

So that the cycle time optimization is efficient, some of the IMM adjustment parameters must be checked or modified :

- the Partial Opening position (OPA) must be adjusted to authorise the robot arm's descent as soon as possible (avoiding any risks of collision)
- the IMM movement speeds and accelerations (mould, ejectors and core pullers) must be optimised to a maximum, whilst at the same time respecting the quality of the parts produced and the safety of the equipment.
- if possible, the fastest movements must be respected : for example, if the mould opening is quicker than the part ejection, the cycle time will be less penalised if the opening is large enough to mask the ejectors and/or core pullers back and forward time during the robot movements.

Reminder : the equipment's productivity is calculated over long production periods. Consequently, stops due to incidents are taken into account.

- ▶ An efficient cycle time optimization **MUST** :
 - Reduce the part fabrication time
- ▶ An efficient cycle time optimization **MUST NOT** :
 - Increase the number of reject parts
 - Increase the number of incidents

Sepro proposes, as an option, an additional means of cycle time optimization : anticipated restart that enables you to mask the IMM reaction time (see chapter V – 5. page 64).








IV – PROGRAMMING INSTRUCTIONS

There are basically four different types of instructions in the main program.

- ▶ The predefined actions for easy control without having to worry about activated physical outputs or their associated checks.
- ▶ Operations concerning output actuation, bit or input tests, binary state tests, counter handling or time delays.
- ▶ Numerical axes' movements.
- ▶ Calls, characterizations and definitions of subroutines.







IV – 1. Predefined actions

IV – 1. 1. Bistable pneumatic commands



















Keyboard selection keys	Corresponding command	Old CN900 version equivalent	Type of action *
	Gripper horizontal	ACT 09	C
	Gripper vertical	ACT 10	C
	Rotation 2 - direction	ACT 14	C
	Rotation 2 + direction	ACT 13	C
	Rotation 2 intermed. position	ACT 16	C
 then <input type="text" value="1"/> to <input type="text" value="8"/> then <input type="button" value="ENTER"/>	Grip part 1 -> 8	ACT 11 ACT 19 ACT 21 ACT 23 ACT 25 ACT 27 ACT 29 ACT 31	C
 then <input type="text" value="1"/> to <input type="text" value="8"/> then <input type="button" value="ENTER"/>	Release part 1 -> 8	ACT 12 ACT 20 ACT 22 ACT 24 ACT 26 ACT 28 ACT 30 ACT 32	C

* Type of action : NC : Not controlled
C : Controlled

Optional second arm :

Keyboard selection keys	Corresponding command	Old CN900 version equivalent	Type of action *
	Pneumatic arm 1 down	ACT 04	C
	Pneumatic arm 1 up	ACT 02	C
	Pneumatic arm forward	ACT 07	C
	Pneumatic arm backward	ACT 08	C
	Sprue grip	–	C
	Sprue release	–	C

IV – 1. 2. Commands for the injection moulding machine

Key	Pulse number	Corresponding command	Old CN900 version equivalent	Type of action
	1st	Ejectors 1 out	SET OUT 30	NC
	2nd	Ejectors 1 out controlled	SET OUT 30 + IN 12	C
	3rd	Stop validation ejectors 1 out	RST OUT 30	NC
 Robot with 2nd IMM	4th	Ejectors 2 out	SET OUT xx	NC
	5th	Ejectors 2 out controlled	SET OUT xx + IN yy	C
	6th	Stop validation ejectors 2 out	RST OUT xx	NC
	1st	Ejectors 1 in	SET OUT 29	NC
	2nd	Ejectors 1 in controlled	SET OUT 29 + IN 11	C
	3rd	Stop validation ejectors 1 in	RST OUT 29	NC
 Robot with 2nd IMM	4th	Ejectors 2 in	SET OUT xx	NC
	5th	Ejectors 2 in controlled	SET OUT xx + IN yy	C
	6th	Stop validation ejectors 2 in	RST OUT xx	NC
  + 	1st	Validation core puller 1_1	SET OUT xx	NC
	2nd	Validation core puller 1_1 controlled	SET OUT xx + IN yy	C
	3rd	Stop validation core pullers 1_1	RST OUT xx	NC
  + 	1st	Validation core pullers 2_1	SET OUT xx	NC
	2nd	Validation core pullers 2_1 controlled	SET OUT xx + IN yy	C
	3rd	Stop validation core pullers 2_1	RST OUT xx	NC
  + 	1st	Robot with 2nd IMM Validation core puller 1_2	SET OUT xx	NC
	2nd	Validation core puller 1_2 controlled	SET OUT xx + IN yy	C
	3rd	Stop validation core pullers 1_2	RST OUT xx	NC
  + 	1st	Validation core pullers 2_2	SET OUT xx	NC
	2nd	Validation core pullers 2_2 controlled	SET OUT xx + IN yy	C
	3rd	Stop validation core pullers 2_2	RST OUT xx	NC
	1st	Machine cycle validation	SET OUT 28	NC
	2nd	Stop machine close command \$	RST OUT 28	NC
	1st	Await end of machine cycle	ACT 00 + SET OUT 28	C
	2nd	Await end of robot cycle if parameter 178 =1	Inexistent	C

* xx and yy represent the number that may vary from one robot to another (see electrical drawing).

Details of the machine cycle commands :

MACHINE CYCLE VALIDATION :

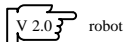
The command is sent to the IMM if :

- ▶ the robot is not executing an end of cycle stop,
- and
- ▶ the part made memory is at 0,
- and
- ▶ the IMM is in automatic or semi-automatic,
- and
- ▶ the robot is in the Arm Free Safety area (SBD), except in the case of an anticipated restart.

AWAIT END OF MACHINE CYCLE :

This command stops the robot, which then waits for the following conditions before moving onto the next step :

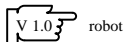
- ▶ mould open,
- and
- ▶ gate closed,
- and
- ▶ IMM in automatic or semi-automatic,
- et
- ▶ part made memory = 1.



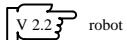
WAITING FOR END OF ROBOT CYCLE : (only accessible if parameter 178 = 1)

This command enables you to stop the robot cycle during an end of cycle request.

See example in chapter V – 7. page 67.



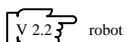
are reset to zero by the Mould Closed input



are reset to zero if the robot goes into fault when the arm is in the mould.


THE CORE COMMANDS :

You must always program the end of the command.




The core commands are reset to zero if the robot goes into fault when the arm is in the IMM.

IV – 1. 3. Other predefined commands

Keyboard selection keys	Corresponding command	Old CN900 version equivalent	Type of action
 2nd pulse	Step duration not controlled	ACT 90	NC

This instruction allows you to momentarily deactivate the watch-dog time between steps.

If the robot is configured for automatic program changing, and if we are in PRG 00 :

Keyboard selection keys	Corresponding command	Old CN900 version equivalent	Type of action
 1st pulse	Await validation PRG change	ACT 99	C

See example in chapter V – 6. page 66.

IV – 2. Instructions

IV – 2. 1. Variables

Name	Mnemonic	Key	Number	Functions
Output	OUT *		000 -> 255	Boolean image of an action (output) to be performed externally (transmitted by the Output board).
Input	IN *		000 -> 255	Boolean image of an information (input) coming from outside (received by the Input board)
Counter	CNT	SHIFT	00 -> 15 0041 -> 9980	Structures reserved for increments and decrements.
Bit	BIT *	SHIFT	000 -> 127	Internal boolean variable
Timer	TIM	SHIFT	000 -> 15	Internal boolean variable resulting from the PLC timer.
Word	WORD	SHIFT	000 -> 4095	General data in 16 bit memory.
Double Word	WWORD	SHIFT	000 -> 127	General data in 32 bit memory.

* The mnemonic code of these structures is also an instruction code.

IV – 2. 2. Boolean instructions

► Temporary output actuation :

Syntax : OUT... (000 -> 255)

- 0 -> 127 : local outputs
- 144 -> 255 : remote outputs (on CAN or ASI network)

The output is actuated during the step ; it is set to 0 when the next step is decoded.

Note : It is not possible to program the “force overtravel”, “Arm Free Safety” (SBD), Without robot and pneumatic High Speed safety outputs.

Note : For a “PIP” robot, the output attributed to the IMM command (parameter : 573) only goes to 1 if :



- The arm is out of the mould,
- and the part made memory is at 0 (last moulded part taken by the robot),
- and the robot is in Step by Step or Automatic mode,
- and the IMM is in Automatic or Semi–Automatic mode.

▶ **Checking an input's status :**

Choice : REVERSE (NORMAL is implicit)

Syntax : IN... (000 -> 255) or IN/....

- 0 -> 127 : local inputs
- 136 -> 143 : pendant inputs
- 144 -> 255 : remote inputs (on CAN or ASI network)

Status “1” (or “0”) of the input is awaited before going on to the next step. Several different inputs can be checked at 0 or 1 in the same step.

▶ **Checking a bit's status :**

Choice : REVERSE (NORMAL is implicit)

Syntax : BIT... (000 to 127) or BIT/....

Status “1” (or “0”) of the bit is awaited before going on to the next step. Several different inputs can be checked at 0 or 1 in the same step.

IV – 2. 3. Allocation and operation instructions▶ CNT instruction – Counter handling –

You have at your disposal :

- 16 standard counters (CNT 0000 -> 0015),
- 4000 stacking counters (see chapter V – 3. page 59).

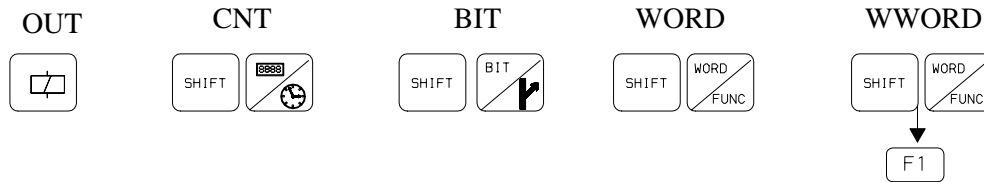
In the main programs, 3 operations can be performed on the counters :

- [RST] Counter reset to zero
- [+ 1] Increment ; $\text{counter}_{(t)} = \text{counter}_{(t-1)} + 1$
- [- 1] Decrement ; $\text{counter}_{(t)} = \text{counter}_{(t-1)} - 1$

**RST.CNT 0001.** Counter No. 01 is set to 0.**DEC.CNT 0015.** Counter No. 15 is decremented.**INC.CNT 0013.** Counter No. 13 is incremented.

► SET instruction – Allocation –

The following variables can be set :



For the CNT, WORD and WWORD, you can choose the following operators :

▪ **The arithmetic operators :**

- = Allocation of a value
 - + Addition
 - Subtraction
 - * Multiplication
 - / Complete division (the remainder is not kept)
- } WORDs or WWORDs are not tested to see if their capacity has been passed.

▪ **The logic operators :**

- = 2 operators are equal
 - >= Greater than or equal to
 - <= Less than or equal to
- } Comparison (IF instruction)
- AND Logic AND
 - OR Logic OR
- } For masking



SET OUT 20. Output 20 is set to 1.

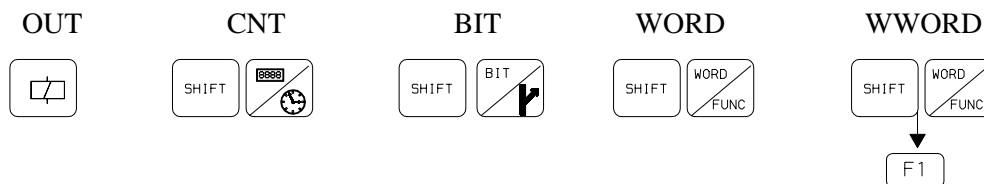
SET CNT 0011 = 0010_D. The decimal value 10 is set in the counter 0011.

SET WWRD 76 + 2. 2 is added to WWRD 76. This is an example of allocation.

SET CNT 7 – CNT 3. The value is given (CNT 7 – CNT 3) to CNT 7.

► RST instruction – Resetting –

The following variable can be set :



RST OUT 20. Output 20 is set to 0.

RST BIT 100. Bit 100 is set to 0.

IV – 2. 4. IF test instruction

This instruction evaluates the variable it contains. Depending on the result, the instruction will or will not execute the next instruction (IF instruction must never be used alone).

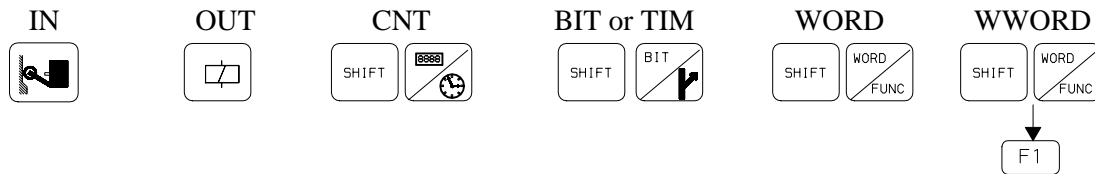
After an IF, all the instructions are valid APART FROM : L, R, MASTER and SLA.

You are offered two choices, once you have selected a variable :

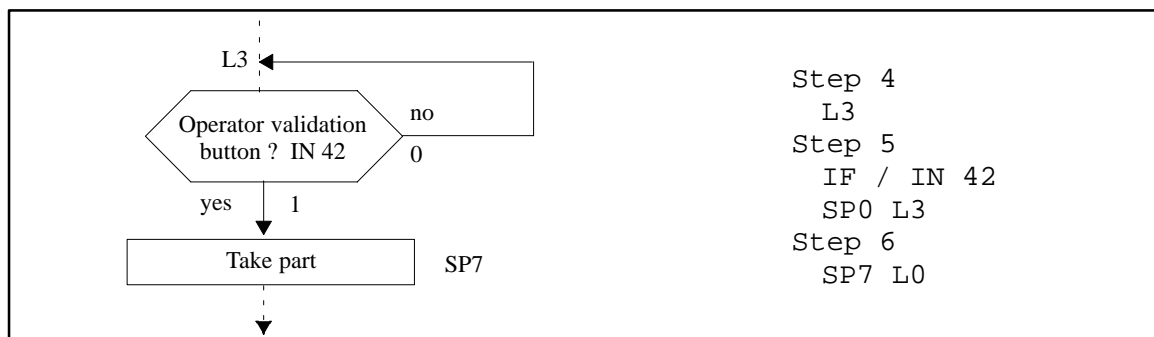
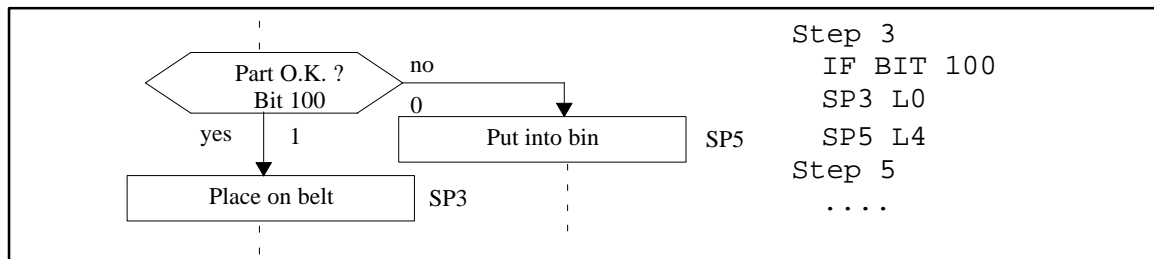
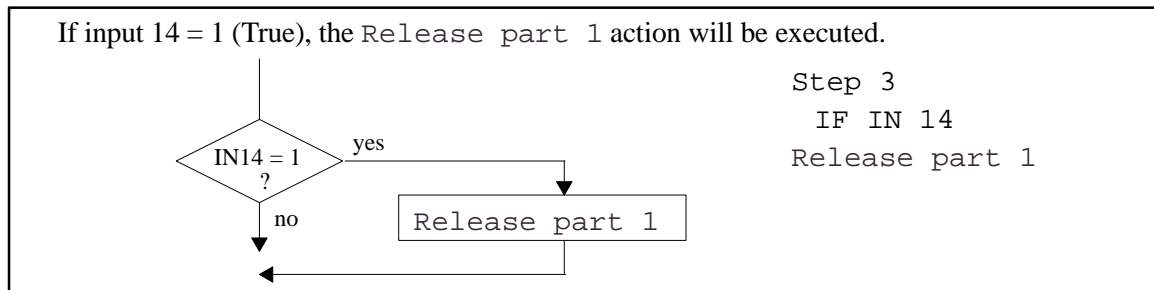
- **IF** : if the condition evaluated is *TRUE*, the next instruction is executed.
- **IF/** : if the condition evaluated is *FALSE*, the next instruction is executed (or if the condition is true, the next instruction is not executed).

Note that these instructions will condition the execution of any subroutine. Complex conditions can be calculated in the PLC and tested in the main program in IF Bit... or IF/Bit... form, followed by the call instruction of the desired subroutine.


The following variables can be checked :



For the CNT, WORD and WWORD, the IF instruction enables you to carry out the following comparisons : =, >= and <=.



IV – 2. 5. Time delays : TIME

Using the  key, the following value can be assigned to the time delay :

- a numeric value from 001 to 999 in 1/10s

Note : This instruction delays running the contents of the step in which it is programmed. If it is the only instruction in the step, it delays the execution of the following step. The outputs programmed in the previous step are maintained during the programmed time.



The following :

```
Step 3  TIME 010  
        OUT 032  
        IN 025
```

has the same effect as :

```
Step 3  TIME 010
```

```
Step 4  OUT 032  
        IN 025
```

Instructions OUT 032 and IN 025 will only be executed after a delay of 1 second.



In the following case :

```
Step 6  OUT 032
```

```
Step 7  TIME 20
```


Output 32 is maintained for 2 seconds when Step 7 is executed.



In the case where a time delay instruction and a special home return request are programmed within the same step, the special home return request will only be taken into account at the end of the time delay.

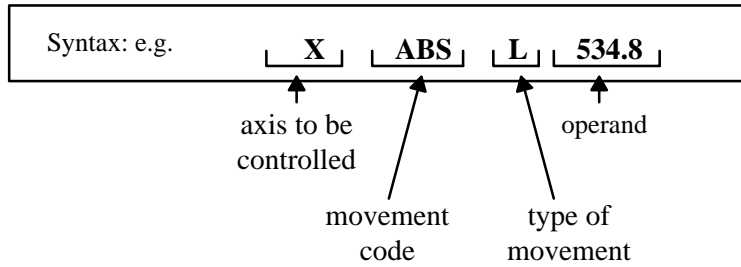


```
Step n  SR 01  
        TIME 20  
        . . .
```

If a home return is requested (press ) in step n before the end of the time delay, SR00 instead of SR01 will be carried out.

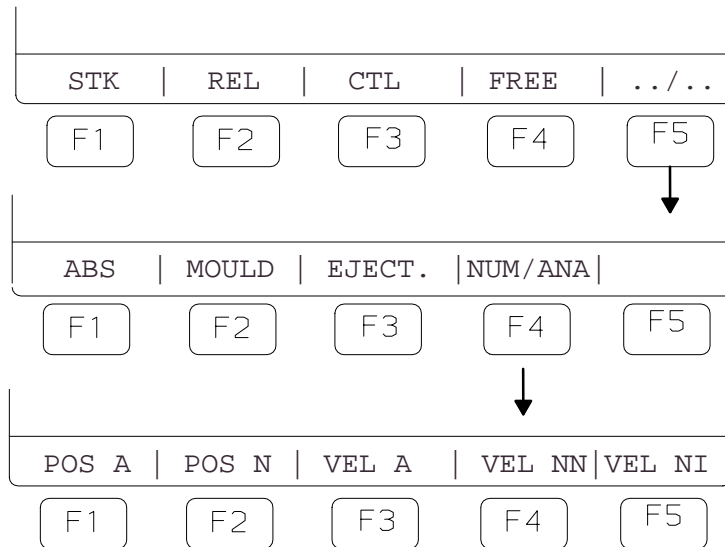
IV – 3. Motorized movement codes

These instruction codes are used to control a movement on a given axis.



IV – 3. 1. Movement code

Having selected the axis to be controlled (if it is motorized) the following movements are proposed :



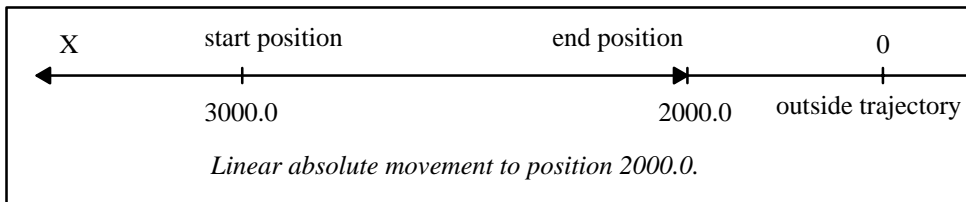
*** ABS : Absolute movement**

An absolute movement is a movement to a given position of the referential. This movement is selected by default. It is therefore possible to enter the operand value (numerical value) directly after selecting the axis. The value will always be positive as 0 is defined outside the trajectory.

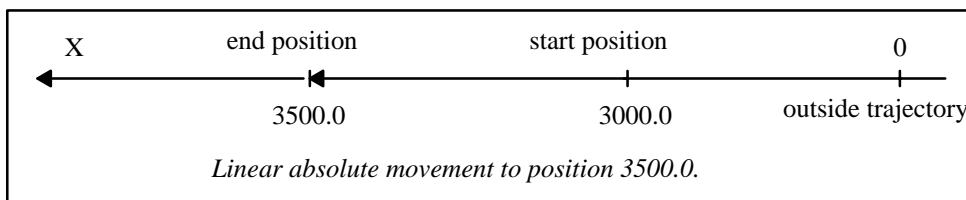
Syntax : X.ABS L position in 1/10 mm (00000.0 to 99999.9)
B. ABS R position in 1/10 degrees (000.0 to 360.0)



X.ABS L 2000.0



X.ABS L 3500.0



*** POSA / POSN / VELA / VELNN / VEL NI : Slaved movements**

Use of these movements is described in the "Mould chasing" manual.

*** STK : Stacking movement**

Use of this movement is described in the S900–II Programming Level 2 Manual. An example is given in chapter V – 3. page 59.

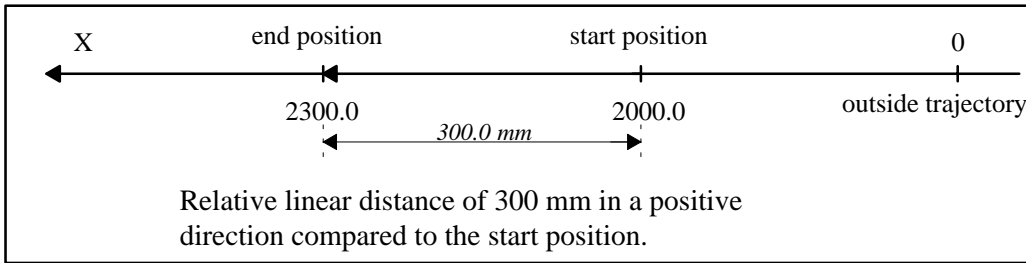
*** REL : Relative movement**

A relative movement is a given movement over a specific distance, compared to a starting point.

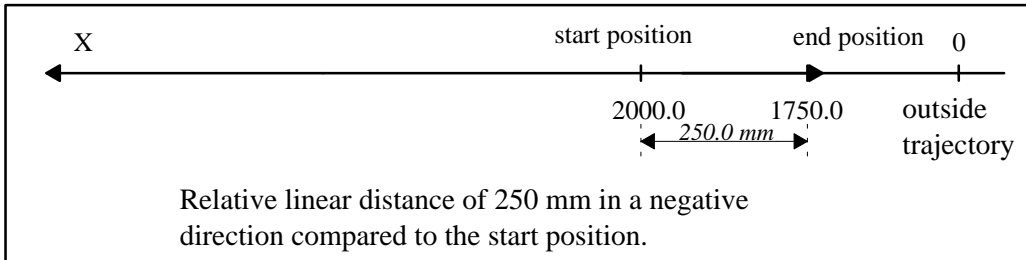
Syntax : X.REL L distance in 1/10 mm (± 0000.0 to 9999.9)
B.REL R distance in 1/10 degrees (± 000.0 to 90.0)



X.REL L + 300.0



X.REL – 250.0



*** CTL : Control movement**

Use of this movement is described in chapter IV – 4. 5. page 48.

*** FREE : Freed movement**

This movement is used to release an axis brake without starting up the motor. Counting continues and provides the position of the robot even if this changes as the result of an outside action (ejector, mould, etc.).

Syntax : Y . FREE

This code is only valid if the requested axis can be freed – marked in the parameters (otherwise, consult our After Sales Service).

This code's action is maintained during the steps following the step where it is programmed until the next movement request for this axis is sent.



Do not release the vertical axes unless a balancing device has been installed.



Step 3	Y . FREE
Step 4
Step 5
Step 6	Y ABS

The Y axis is free during steps 3, 4 and 5.

IV – 3. 2. Type of movement

▶ L = Linear

▶ R = Rotary

This information appears automatically when a motorized movement is programmed. It corresponds to the type of movement related to the axis and it is provided in the axes' definition parameters.

IV – 3. 3. Operand

After selecting the movement code :



The operand is immediately given the value of the axis' current position (if the axis is intialized).



The X axis is at the position X = 2317.4

. Press  then [. . / . .]  then [ABS] .

-> the robot displays > X ABS L 2317.4 (the operand flashes).

. Press  twice to confirm this value or enter a new value using the alphanumeric keys then confirm by pressing .

-> the robot displays X ABS L 2317.4



By switching from programming to adjust mode, it is possible to move the robot's axes. When you select programming mode again, you go back to the program step from which you left. This procedure enables you to teach the position values whilst entering the program.

The operand can be a numeric value (in 1/10 mm or in 1/10 degrees).

One of the following functions can be used :










* **wword** : 32 bit word

Use of this operand is described in the S900–II Programming Level 2 Manual.

* **Teach** : Teaching

The operand will only be assigned a value when the program is executed.

When the robot carries out a Step containing a programmed movement whose destination was declared in "teaching" mode, a message informs the operator that he must move the axis himself using the       keys. The final position of the axis is validated by the operator by pressing .

Syntax : X.ABS L Teaching
 B.ABS R Teaching

Codes and movements which can be taught : ABS, STK, REL and CTL*.

* It is only possible to teach the CTL trigger movements in the SAP source programs.

*** Offset**

Use of this function is described in the “Mould chasing” manual.


*** Val Stk : Position of the first part in a general stacking sequence**

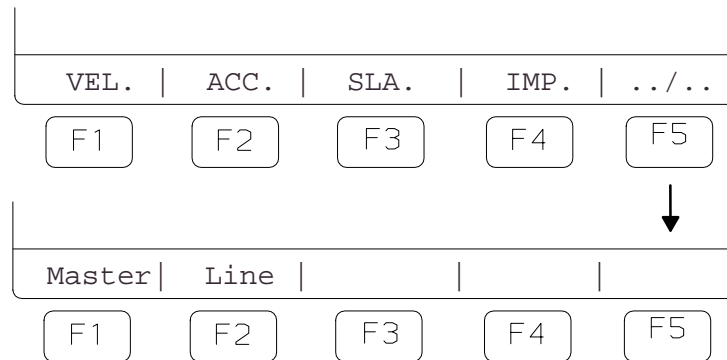
Use of this operand is described in the S900–II Programming Level 2 Manual.

IV – 4. The preparatory functions ”FUNC” of the numeric axes

These are preparatory functions relating to numerical movements. There are basically two types :

- ▶ Temporary–effect functions (valid only for the current step)
- ▶ Maintained–effect functions which are valid until a new function appears.

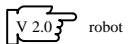
Accessible using the  key :



IV – 4. 1. VEL : Speed axis in% (Maintained)

Used to change the speed of the programmed axis from 1 to 100 (% of the maximum speed).

Note : Speed changes can be programmed as triggered actions in a master movement.



This command can effect the speed and the acceleration in the SAP programs at the same time. For this to be possible, parameter 435 must be at 1.

IV – 4. 2. ACC : Axis acceleration in % (Maintained)

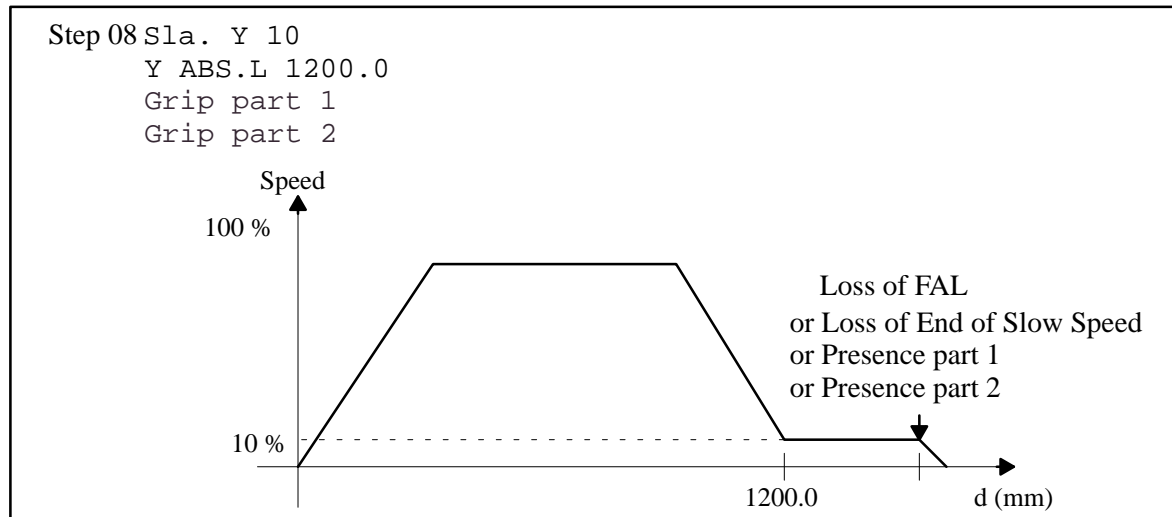
Used to change the acceleration of the programmed axis from 1 to 100 (% of the maximum acceleration).

Note : Acceleration changes can be programmed as triggered actions in a master movement.

IV – 4. 3. SLA : Slow approach (Temporary)

Syntax : SLA Y 15
 SLA Z 20

Use : this type of function is related to the execution of a numerical movement.



In the above example, the Y movement will normally be run to reach a speed of 10% at the position 1200.0. Subsequently, the speed will be maintained until :

- ▶ “End of Slow Approach” (FAL) input disappears. (Generally on input of the optional changeable parallelogram setting control on SEPRO wrists). Signal active at 0.
- ▶ or the disappearance of the “End of Slow speed” input. Signal active at 0.
- ▶ or the appearance of all the Part Presence controls whose instructions are programmed simultaneously at the slow approach, signal active at 1.



If several grips are programmed, the slow speed will be maintained until all the part present messages appear.

Moreover, a so-called “slow” input is associated with each axis which causes a controlled reduction in speed when it disappears, so that the movement ends as shown above.

Possible uses :

- ▶ Without an external speed–decrease sensor :
 - gripping a part whose position is only vaguely known.
 - lowering a part on to a stack whose height is not known exactly.
- ▶ With an external speed–decrease sensor (direct–reflection cell for example) :
 - gripping (releasing) a part whose axial position is not known but that has to be dealt with rapidly.

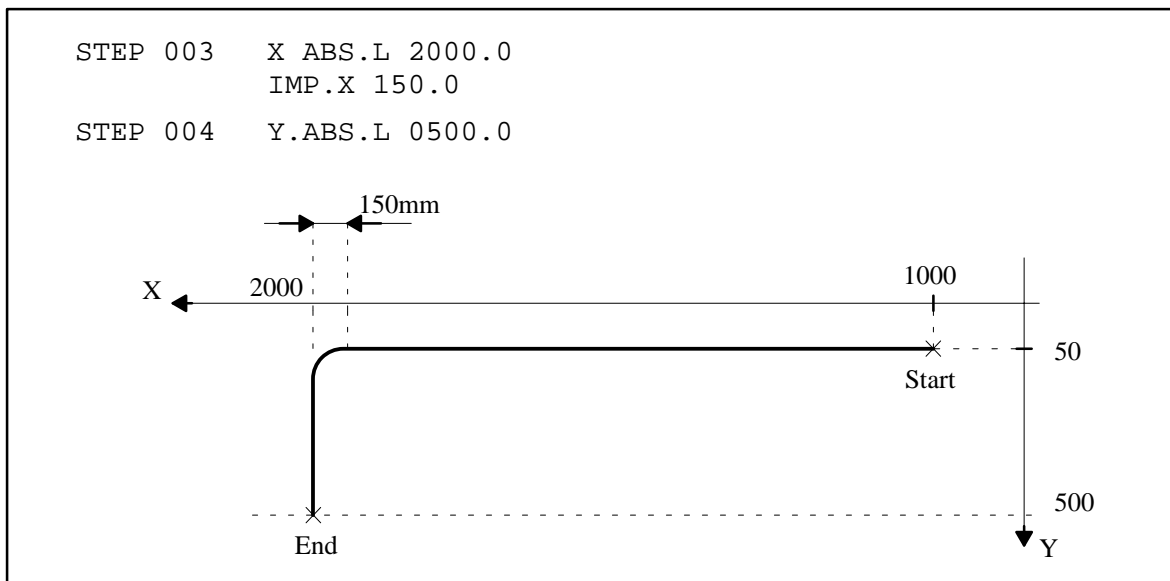
IV – 4. 4. IMP : Imprecision (Temporary)

This function is used to temporarily change an axis' stopping tolerance so that the following step can be started without waiting for the end of the movement declared as imprecise.

The main use of such a function is to gain cycle time.

The step–to–step transition is masked in the end of the movement declared as imprecise.

The programmed imprecision value (3.0 to 999.9mm) must not be greater than the movement wanted. If this happens, the movement will not be executed in automatic mode without triggering a fault signal.



IV – 4. 5. MASTER : Master movement (Temporary)

This function enables a program step to be sub-divided into “sub-steps” separated by control points that can be controlled or triggered by numerical actions or movements.

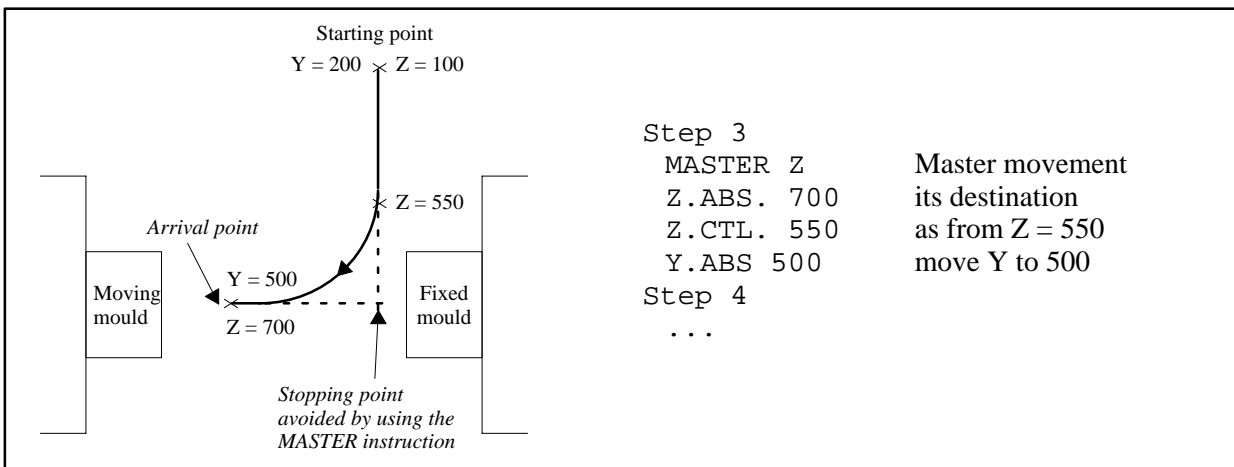
Syntax :

```

MASTER [ ]      <- master movement axis
[ ] ABS [operand] <- master movement destination
[ ] CTL [operand] <- trigger point on master movement (as from ...)
[ ]            <- triggered instruction (... to be done)
    
```



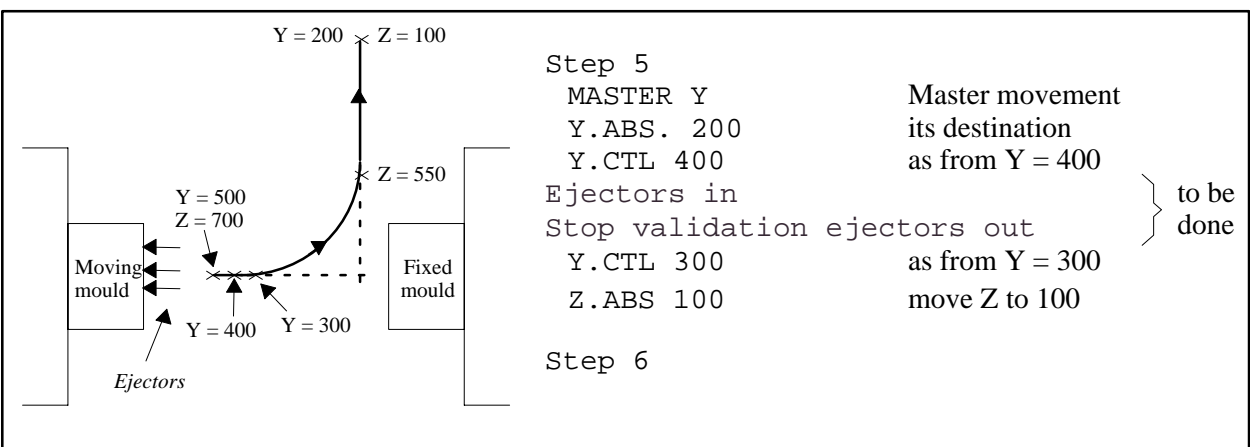
Here is an example of the robot arm accessing the mould, where the advance is triggered during the descent.



Several trigger points are possible in a MASTER instruction.

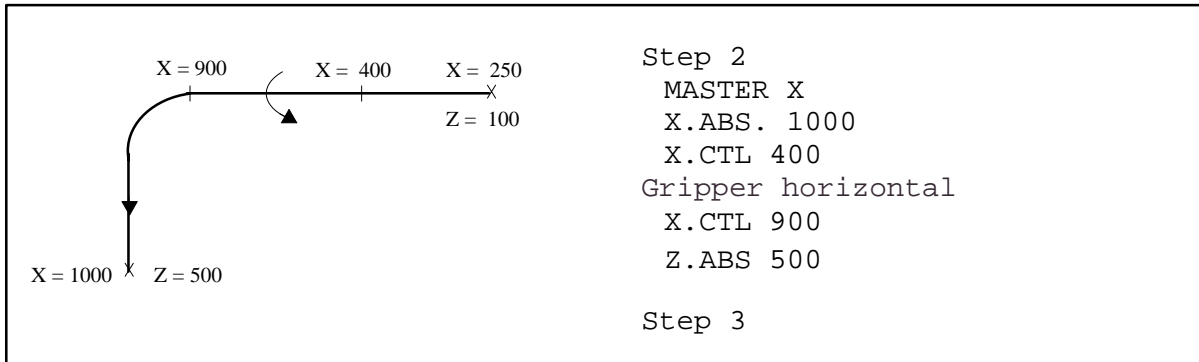


Example of the arm exiting the mould with the ejectors moving back and the arm moving up and back at the same time.



The coherence of the trigger points (sequential order as well as the position compared to the starting point and the arrival point) is monitored by the software. If they are incoherent, the fault D_63: \$ -MVT TO BE MADE OUTSIDE LIMITS appears on the screen.

The controlled predefined actions (see chapter IV – 1. page 32) should be finished before the next CTL instruction, otherwise the “The info to be controlled during the movement is faulty” message appears on the screen.



If the robot arrives at 900 and the horizontal rotation has not finished, the robot goes into fault. To avoid this, replace the predefined action with the associated output command (OUT). See the robot parameters in file S of the Instruction Manual or the electric drawing.

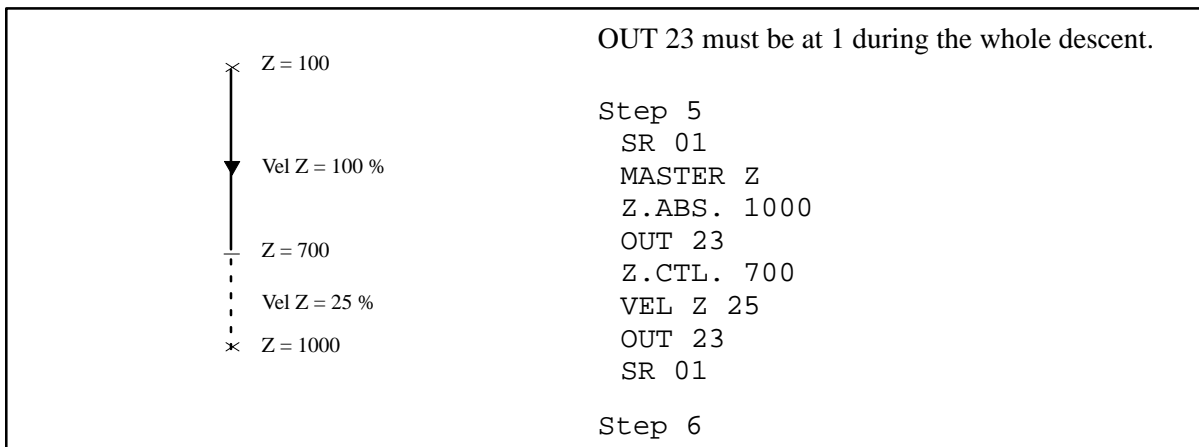
Note : Control point “sub-steps” have the same effect as program steps in relation to : temporarily-actuated outputs, Home returns to be carried out.

In order to actuate an output during the entire master movement, it should be actuated after each programmed control point.

The same applies to Home Returns to be executed.



To keep the outputs activated or to specify a home return, the instructions must be repeated at each trigger point.



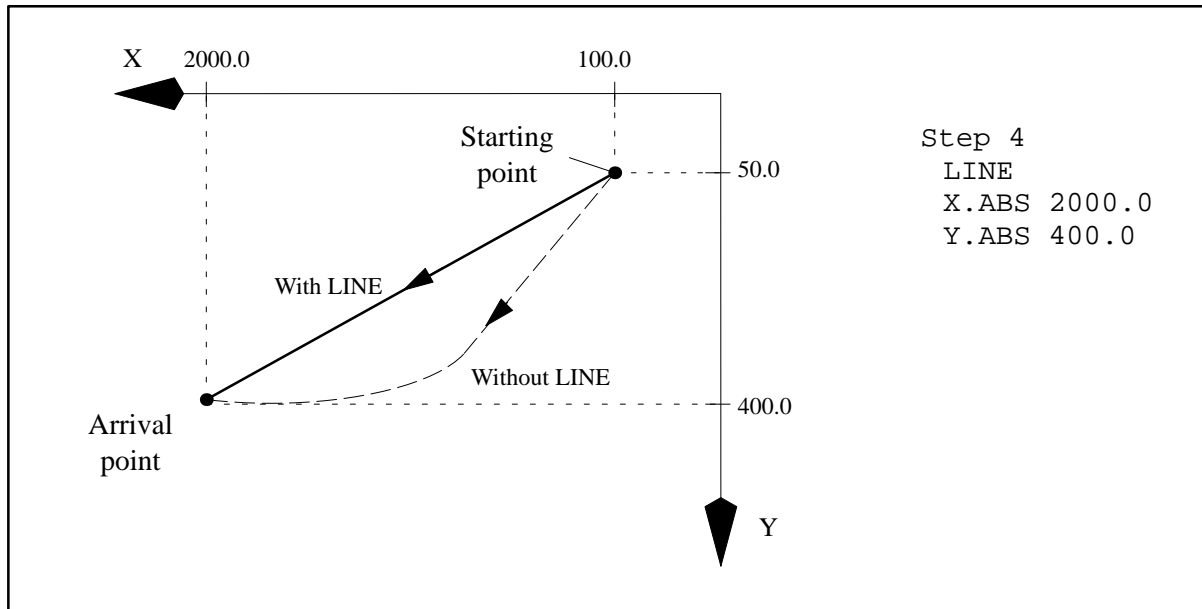
IV – 4. 6. LINE : Linearity (Temporary)

This function is used to program a movement linearity instruction into a Step; i.e. all the axes programmed in the Step containing the LINE programming will terminate their movements together.

Note : LINE does not work with the ABS.L.WWRD and STK.L.WWRD movements.



X and Y movements are to be started simultaneously starting at points X=100, Y=50.0 to go to points X=2000.0, Y =400.0.




Note : Depending on the type of motorization, the path obtained will not be a perfectly straight line.

IV – 5. Specific codes

IV – 5. 1. SP code as an instruction

For its use, see chapter III – 2. page 23.

Accessible using the  key :

Syntax : SP nn L00 (to 99)

After an SP request (instruction), the operands requested are :

- ▶ a number from 00 to 99.
- ▶ A label number, corresponding to a return step indicator and a suffix indicating the order in which the subroutines are performed. When an SP is called up with a return label L00, you will return back to the step following the one where the SP was called.

Note : When one subroutine calls another subroutine, only 3 overlapping levels are possible.

IV – 5. 2. PLC code as an instruction

The PLC is a programmable logic controller which can be associated with a main program if necessary. Its functions are described in the Programming Manual Level 2. See example in chapter V – 1. page 53.

Syntax : PLC 00 (to 98)

Note : The number 00 means that there is no PLC associated with the main program.

IV – 5. 3. SR code as an instruction

For its use, see chapter III – 3. page 26.

Accessible using  then  :

Syntax : SR 01 (to 99)

Note :
. SR 00 is not valid, it is implicit (executed by default).
. When an SR is executed, it returns to the last Rnn Label with the same number as itself.
Example : SR 05 returns to the last R05 encountered before its execution.

IV – 5. 4. “L” and “R” labels

Accessible using  then or :

Syntax : L01 to L99

Syntax : R00 to R99

Note :

- . The label L00 is not valid.
- . The “L” labels are unique, the “R”s can be multiple.
- . They can ONLY be used in the MAIN program, (therefore not in the SP, SR etc....).

(See paragraph III – 2. page 23).

IV – 5. 5. END of PRG, SP..., SR, PLC codes

Syntax : END

This code is automatically added to the end of each PRG, SP, SR et PLC when is pressed. You cannot delete it.



It is always possible to insert steps before the step containing the word END.

V – SPECIFIC PROGRAMMING

V – 1. PLC and parallel subroutines – SPP examples

V – 1. 1. Managing a timed belt indexing

An output activates the belt indexing : .output at 0 -> belt is stopped,
.output at 1 -> belt is indexed.

Solution using a PLC :

- Define the PLC in step 0 of the main PRG (example : PLC 01).
- Start the belt indexing in the main PRG (after the part release).
(Example : SET OUT 20).
- Write in the PLC 01 :

```
IF OUT 20          (output actuating the belt indexing)
TIMER 00
VAL 50            (length of movement in 1/10s)
IF TIM 00
RST OUT 20       (output actuating the belt movement)
```

Solution using an SPP :

- Call-up the SPP in the main program (after the part release).
(Example : SP 81 L0).
- Write in the SPP :

```
Step 0  OUT 20  (output actuating the belt movement)
Step 1  TIME 50 (length of the movement in 1/10s)
Step 2  END
```

V – 1. 2. Managing a belt’s “step by step” movement

An output actuates the belt indexing : output at 0 -> the belt is stopped,
output at 1 -> the belt is indexed.

The release area on the belt is controlled by cell “1”.

Cell “2”, whose position is marked in figure 1, defines the step between the parts.

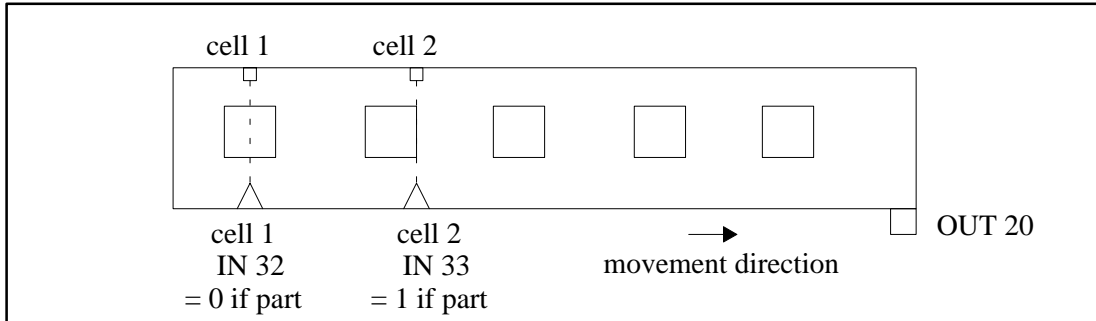


Figure 3 : Conveyor belt (overhead view)

The part must be released in front of the cell so that a part is not released on top of another one.

Solution using a PLC :

- Define the PLC in step 0 of the main PRG (example : PLC 02).
- Start the belt movement in the main PRG (after part release).
(Example : SET OUT 20).
- Write in the PLC 02 :

```

IF /IN 33          cell 2
SET BIT 80         part removed from cell 2

IF BIT 80          part removed from cell 2
IF IN 32           cell 1
IF IN 33           cell 2
RST OUT 20        index belt output

```

Solution using an SPP :

- Call-up the SPP in the main PRG (after part release).
(Example : SP 82 L0).
- Write in the SPP :

```

Step 0  OUT 20  index belt output
        IN/33  cell 2
Step 1  OUT 20  index belt output
        IN 33  cell 2
        IN 32  cell 1
Step 2  END

```

V – 1. 3. Maintaining a pulsed input

An input from the S900–II numeric control can change status at any point during the robot cycle. Its function can be, for example, a quality control request, a “bad part” data item, an access request....

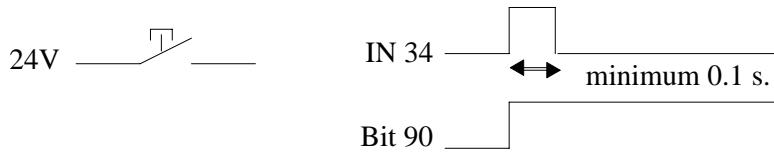
These data items are nearly always asynchronous to the robot’s sequential cycle. It is therefore necessary to retain them so that they can be used at a strategic point in the robot’s cycle.

A pulsed input can only be maintained with a PLC.



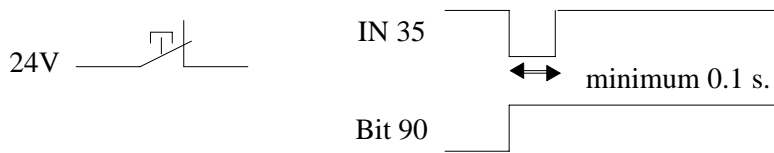
It is however possible to maintain one of the customized keys, without using the PLC (see example below).

Maintaining an input set to 1 :



```
IF IN 34
SET BIT 90 -> Bit 90 is set to 1 after
input 34 has been set to 1.
```

Maintaining an input set to 0 :



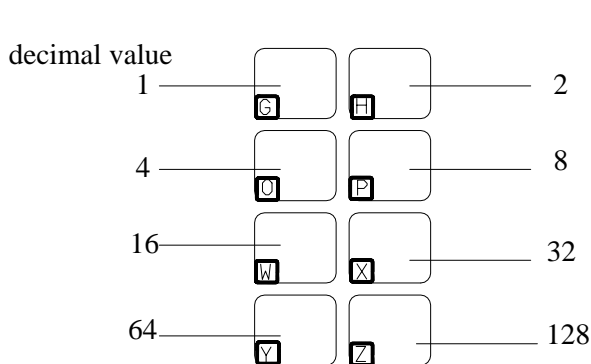
```
IF IN 35
SET BIT 90 -> Bit 90 is set to 1 after
input 35 has been set to 0.
```

This bit 90 will need to be used in the main program. However, do not forget to set it to 0 after having used it with a RST Bit 90.



Memorising a customized key pulse :

The 8 customized keys can be used to memorise an operator request. Pressing one of these keys forces WWRD 120 to the value corresponding to the code of the last key pressed. You just have to test the value of WWRD 120 when needed in the program then reset it to zero after you have used it.



```
IF WWRD 120 = 1 G key pressed
SP 10 L1 SP to be run
Do not forget to reset WWRD 120 to 0 in
SP 10
```

V – 2. System data items

In order to switch to another part of the program, it may be necessary to call–up internal data items from the S900–II numeric control.

V – 2. 1. System bits

Bits 0 to 33 are reserved and set by the S900–II numeric control.

Bits for the regular stacking sequences :

- Bit 0 = 1 if reverse stacking
- Bit 1 = 1 if end of layer or end of row
- Bit 2 = 1 if end of pallet
- Bit 3 = 1 if odd stack in progress
- Bit 4 = 1 if odd column in progress
- Bit 5 = 1 if odd layer in progress
- Bit 6 = 1 if odd part in progress (Stacking SP counter)
- Bit 7 = 1 if stacking sequence in progress.



When the end of pallet bit (bit 2) is actuated, the end of layer or row bit (bit 1) is not active.

These bits are set to 1 or 0 as soon as the stacking subroutine header has been decoded.

Bit 7 is set to zero when the END instruction of the stacking subroutine is decoded or if a subroutine, whose return label L is other than 0, is started.

The other bits (0 to 6) keep the same status until the next stacking subroutine heading is decoded.

These operations can be conditioned by the system bits.



- 1) IF BIT 2 if end of pallet
 SP 81 L0 start the execution of SPP 81

- 2) IF BIT 3 if odd stack in progress,
 Rotation 2 – direction turn the part

You do not have to set bits 0 to 7 to zero again; this is done automatically by the S900–II control.

Robot's status bits:

Three system bits enable the organisation of the program architecture.

- Bit 9 = 1 if a TOTAL home return is running.
- Bit 10 = 1 if a parallel subroutine SPP is running.
- Bit 11 = 1 if an end of cycle triggered by the Without Robot mode is running.

Note : The robot's cycle end can be tested using the weight 2 bit of word 36 (IF WRD 36 AND 2).

Using bit 9

When a Total home return is executed, the stacking counters are set to 0. Therefore, the container must be removed before another one is put into place.



- ```

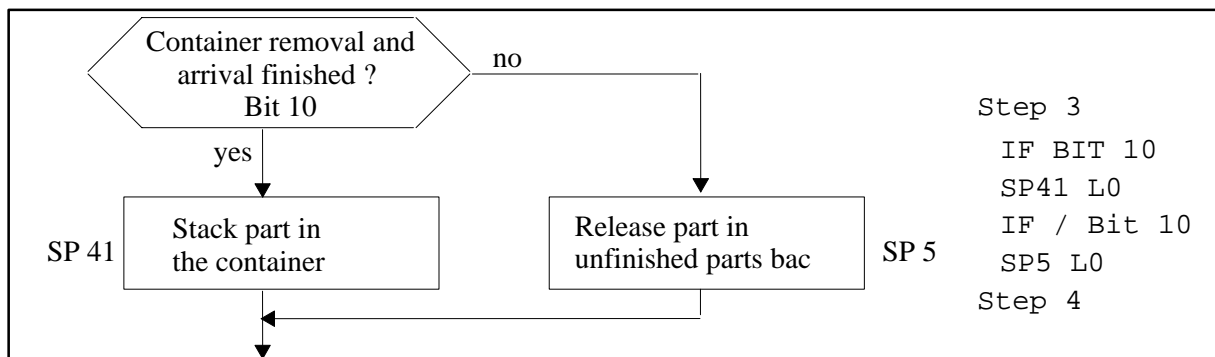
1) SR xx
 IF BIT 9 if total home return is running
 SP 81 L0 remove the container

2) SR yy
 IF BIT 9 if total home return is running
 SET Bit 55 removal / new box request




```

### Using bit 10


It is not possible to actuate two SPPs at the same time or restart an SPP which is already actuated. It is therefore possible, using bit 10, to check whether an SPP has finished before starting another one.

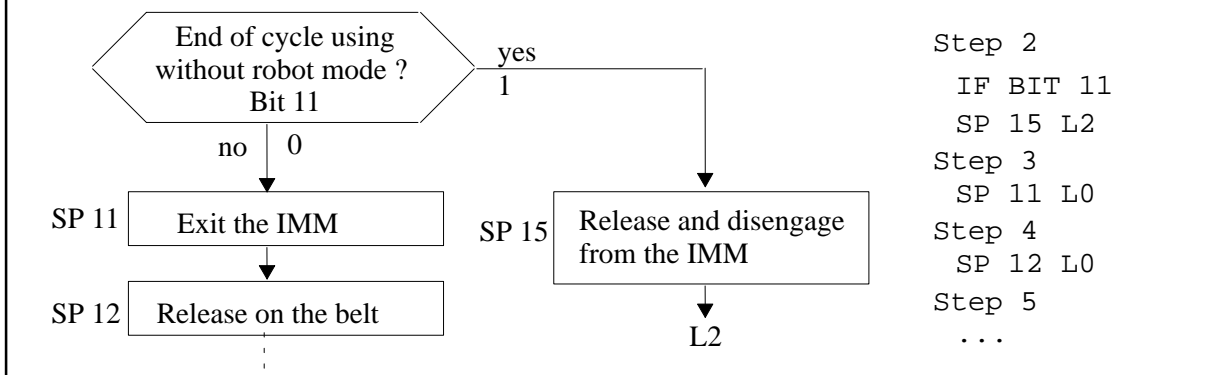


### Using bit 11

When you request a robot stop (robot OFF ) , the robot carries out an end of cycle. Bit 11 is used to differentiate between the real end of cycle  and that triggered by selecting robot OFF .



You wish to stop the robot just after the part grip when you select robot OFF  . Therefore, you decide to leave the part in the mould and not to release it elsewhere. In the part grip SP, you must write :



The other system bits (bits 12 to 33) are described in the Programming Level 2 manual.

**V – 2. 2. The part counters**

A certain number of counters can be viewed in the production pages.

**The part counters**

. The number of **parts finished**, which can be viewed in the parts page of the production menu, is generated in the program.

For the counter to evolve, it is necessary to program the increment, i.e. : SET **WWRD 76** + xx\_D where xx equals the number of parts per mould.

. The number of **parts to be done**, which can be viewed in the parts page of the production menu, can be modified using the two following methods :

- directly in the consultation page (see S900–II User Manual),
- by the program, where the counter is **WWRD 68** : SET WWRD 68 = xxxx\_D and where xxxx is the number of parts to be made.



. Using the parts counter :

- 1) Trigger an end of cycle stop when you arrive at the number of parts to be done.

Write in the program :

```
IF WWRD 76 >= WWRD 68
SET WRD 59 OR 2_D
```

- 2) Carry out a quality control when you arrive at the number of parts to be done.

Write in the program :

```
IF WWRD 76 >= WWRD 68
SP 15 L0
```

where SP 15 describes the part release at the quality control station.

**The counters**

The counters 0, 1 and 2 can be consulted in the counter page of the production menu.

The counters must be assigned values in the programs.



- 1) INC CNT 0
- 2) SET CNT 1 = 3\_D
- 3) RST CNT 2

### V – 3. Example of part palletization

Subroutines 41 to 60 are used to easily define part palletization sequences. They have a sequential structure, but contain a header area as well as steps 0 to 999. This header describes the organization of the pallet (number of parts per axis, gap between parts on each axis).

Each time a stacking subroutine is run, the counter associated with it is incremented. When it is equal to the number of parts to be stacked, the counter goes back to 1 for the next cycle.

This counter enables the robot to calculate the position of the part that it must palletize. The number of the counter associated with the stacking subroutine is formed from the program number and the stacking subroutine number.



The counter of stacking subroutine number 41 of program number 15 is : CNT 1541.

The stacking counter is set to 0 after a TOTAL home return. A simple home return has no effect on the stacking, unless you have programmed otherwise.

#### ► Syntax for calling-up a stacking SP :

SP 42 N Lx or SP 42 R Lx (N means Normal and R means Reverse).

Selecting “Reverse” means that the program is executed in the opposite way to that which was initially defined. In “normal” execution, the stacking counter is incremented before the gaps are calculated. In “reverse” execution, the gaps are calculated before the counter is decremented.

#### ► Specific instructions for stacking SPs

So that the axes’ movements in the stacking SPs take into account the gaps between the parts and the stacking counter, you must use the following instructions :

```
X. STK xxxx
Y. STK xxxx
Z. STK xxxx
```

where xxxx are the absolute values defining the position of the first part when the stacking counter equals 1.

The “end of layer or column” and “end of pallet” commands in the stacking SP header are :

- actuated as soon as the stacking SP is run,
- memorized after the stacking SP has been run.

Therefore, don’t forget to set these commands back to 0 once they have been used in the program, if necessary.



When the “end of pallet” command is active, the “end of layer or column” command isn’t.

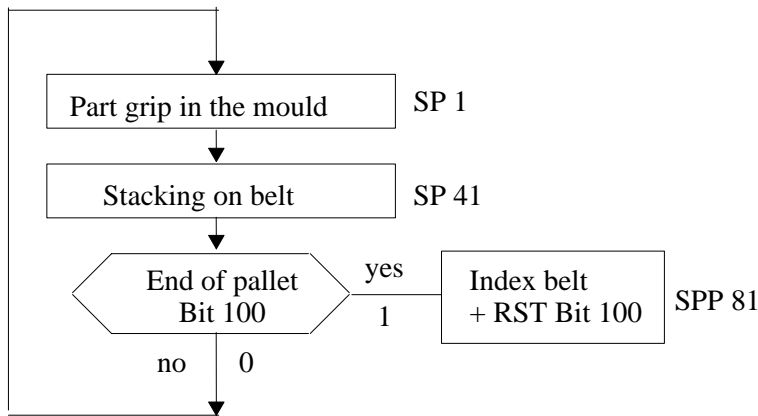
Two specific operations for the stacking subroutines are available :

- INC.STK          Increment the counter of the stacking subroutine running.
- DEC.STK          Decrement the counter of the stacking subroutine running.

These two instructions do not have a number operand. They will only be used in the home return subroutines SR. The counter that is incremented or decremented will be that of the subroutine where the SR is requested.



The main program can be written as follows :



```

PRG 1
Step 0
 PLC 0
 SP 1 L0
Step 1
 SP 41 N L0
Step 2
 IF Bit 100
 SP 81 L0
Step 3
 END

```



A simple stacking routine :

SUBROUTINE-> REGULAR STACKING 41

```

* [
*
Staggered no (0) or yes (1).....0
By layers (0) or stacks (1).....0
Stacking (0) or unstacking (1).....0
Storage in XY (0) or YX (1).....0
Number of parts in X row 1.....03
Gap X between parts row 1.....-0100.0
Number of parts in Y column 1.....02
Gap Y between parts column 1.....-0200.0
Number of layers.....03
Gap between layers.....0050.0
Command at layer end or stack end....BIT 000
Command at pallet end.....BIT 100

```

Contents of SP 41 steps

```

Step 0
 X.STK._xxxx
 Y.STK._yyyy
Step 1
 Z.STK._zzzz
Step 2
 Release part
Step 3
 Z.ABS.L 100 Z arm up
Step 4
 END

```

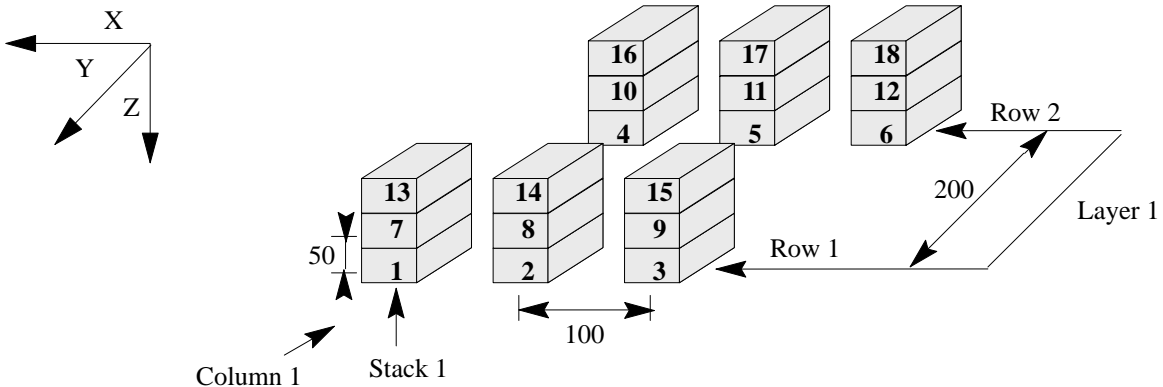


Figure 4 : Stacking or palletization

## V – 4. Customized messages




### V – 4. 1. The comment files

A comments file created with Sepro AS900–II editor for PC can be associated with each main program.

This file can contain comments on :

- ▶ the inputs,
- ▶ the outputs,
- ▶ the bits,
- ▶ the predefined actions,
- ▶ the SAP markers.

These comments can be seen :

- ▶ In “programming” mode :
  - 32 characters maximum for the predefined actions,
  - 26 characters maximum for the comments.
- ▶ In “monitor” mode, once you have pressed  – 33 characters maximum.
- ▶ In “automatic”  or “Step by Step”  mode if the robot is waiting (before the fault “D\_10 Safety time between steps elapsed”) – 33 characters maximum.
- ▶ In the faults concerning the inputs or bits with comments – 33 characters maximum.
- ▶ In the print–out of programs from the robot – 33 characters maximum.

The comments shown come from :

- ▶ if it exists, the comments file of the main program PRG,
- ▶ otherwise, the SAP messages (for the predefined actions),
- ▶ otherwise, the default messages contained in the robot’s memory.

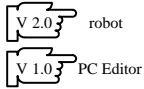
There is no comments file for the PLCs. The comments shown come from :

- ▶ the first main program using this PLC if it has a comments file,
- ▶ otherwise, the default messages contained in the robot’s memory.

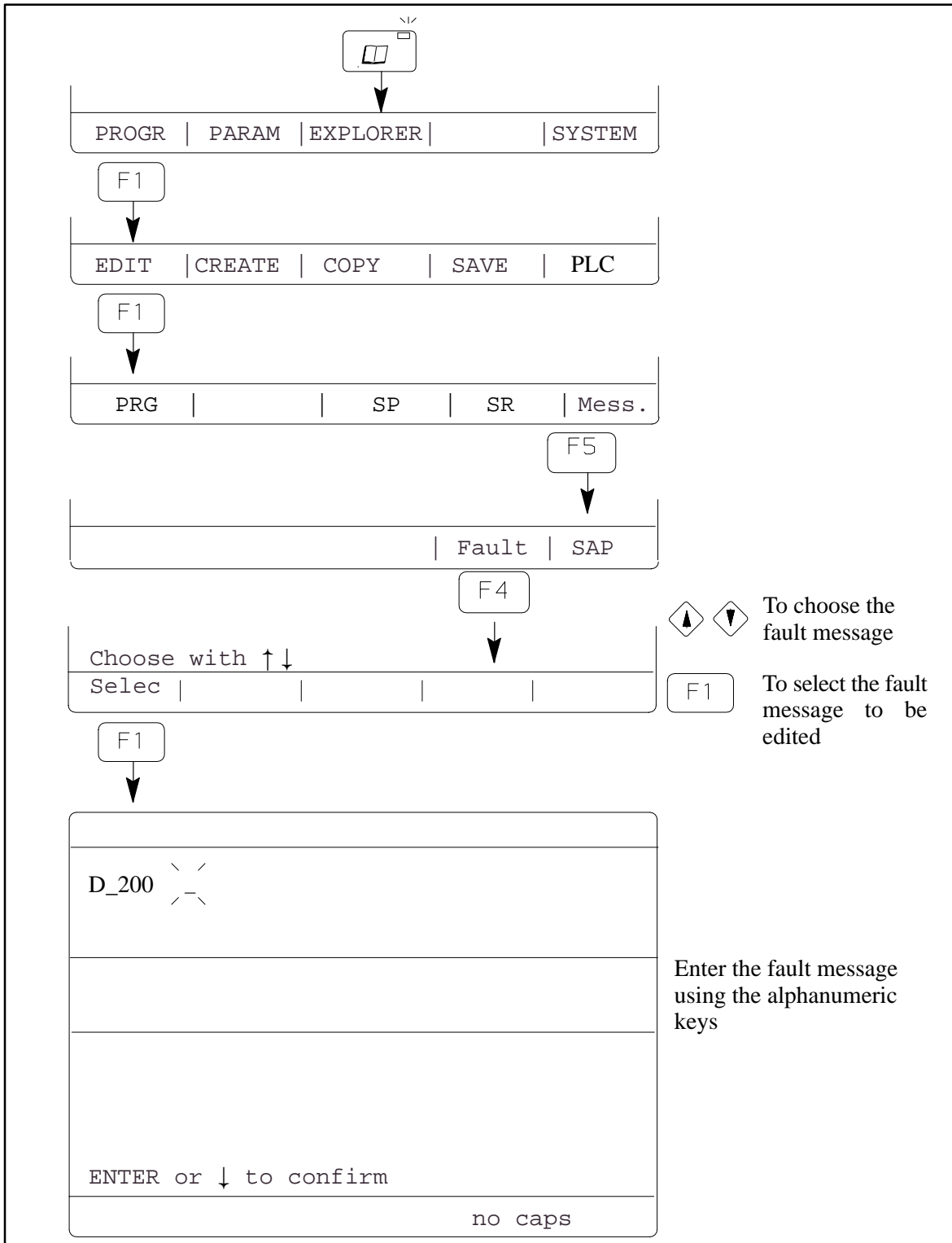
**V – 4. 2. Customized fault messages**

Five faults can be generated per program. These are faults D\_200 to D\_204. A message is associated with each one of the faults, which can be entered or modified in programming mode on the robot.

These messages can be created and modified on the PC. They can also be transferred from the PC to the robot and vice-versa.



Accessing and creating fault messages D\_200 to D\_204 :



Generating a customized fault in automatic mode :

Programming the instruction SET WORD 62 = xxxx (with  $200 \leq \text{xxxx} \leq 204$ ) will put the robot into fault and the corresponding fault message is displayed, when it is executed.



Input 35  
(Box in place)

D\_200 : Box full  
D\_201 : Belt busy  
D\_202 : . . . .  
D\_203 : Box not in place  
D\_204 : . . . .

...

```
Step 3 IF /IN 35
 SET WORD 62 = 0203
```

Step 4  
...

When step 3 is executed, if input IN35 is at 0, the robot goes into fault and displays the message “Box not in place”.

---

Note : For faults 200 to 204, no help message  is available.

---



Pressing  starts the robot up again. Therefore, if necessary, you must program the robot to wait for the cause of the fault to disappear.



```
Step 3
 IN / 35
 IF /IN 35
 SET WORD 62 = 203
Step 4
```

## V – 5. IMM anticipated restart (option)

### Aim :

Shorten the cycle time by masking the IMM reaction time (time between the closing authorization from the robot and the actual mould movement).

### Principle :

The machine cycle validation (VCM) and the arm free safety (SBD) are given when the robot is still inside the mould.

### Safety :

If one of the data items mould open (MO) or partial opening reached (OPA) disappears whilst the robot is still inside the mould, the robot goes into fault and immediately interrupts the mould closing authorization commands.

D\_5 : MOVEMENT OUTSIDE CAMS : if there is no anticipated restart running.

D\_32 : PREMATURE MACHINE RESTART : if an anticipated restart is running.

D\_35 : ANTICIPATED RESTART NOT CONFORM : if the monitoring circuit has been set off.

### Type of anticipated restart :

There are two types of anticipated restart :

- anticipated restart with a programmed delay,
- auto–adaptative anticipated restart.

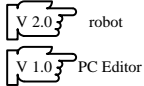
Parameter 174 (RLCE\_ATCP\_1) defines the type of anticipated restart. See “S900–II Configuration” manual.

### Conditions :

The IMM anticipated restart is only effective if :

- the robot is in automatic mode,
- and if ▪ the overall speed coefficient  $K_v = 100 \%$
- and if ▪ the offset wait is not valid (parameter 451)
- and if ▪ the instruction SET WWRD 63 = xxxx is in step 00 of the PRG
- and if ▪ the value of WWRD 63  $\geq \frac{\text{Parameter 175}}{2}$

Case of anticipated restart with programmed delay only



### V – 5. 1. Anticipated restart with programmed delay (Parameter 174 = 2)

For the anticipated restart with programmed delay to be effective, step 00 of the program used must contain the instruction SET WWRD 63 = xxxx where xxxx is the length in 1/10 s. of the delay that you wish to apply between :

- ▶ the decoding of the instruction “machine cycle validation” (VCM),

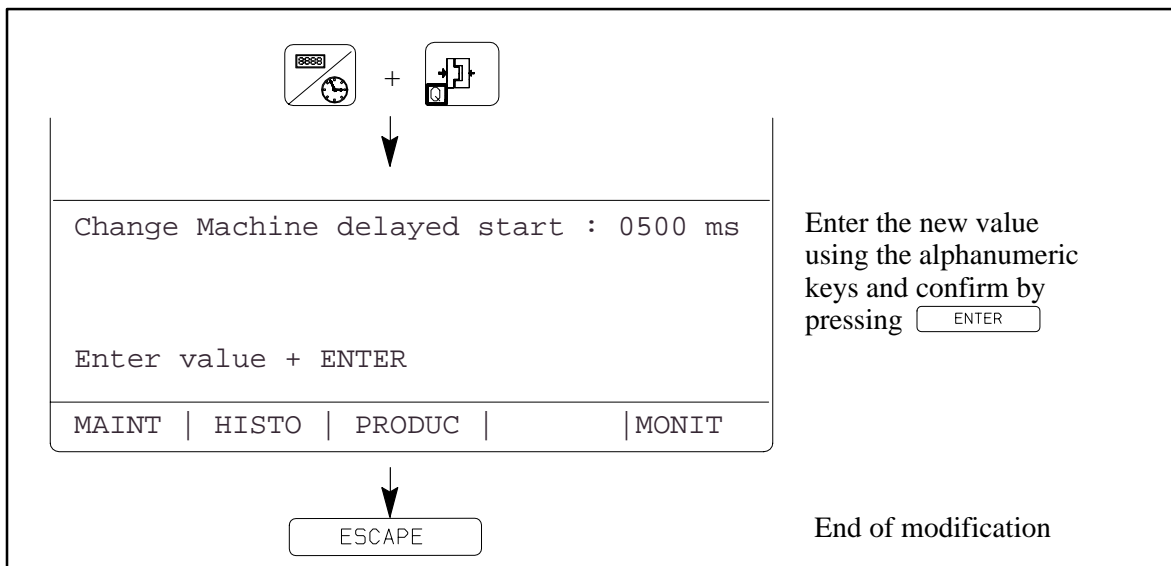
and

- ▶ the activation of the mould closing authorization commands.

Reminder : xxxx must also  $\geq \frac{\text{Parameter 175}}{2}$




To optimize the anticipation, it is possible to change the length of the delay (value xxxx of WWRD63) in automatic mode.



### V – 5. 2. Auto–adaptative anticipated restart (Parameter 174 = 1)

In this case, the robot calculates by itself the delay applied between the decoding of the “machine cycle validation” instruction (VCM) and the activation of the mould closing authorization commands.

The value of the minimum delay that can be applied is in parameter 176. Each time the robot  or the IMM) stops, the delay applied takes the value of parameter 175 again. The calculation is made by successive tries. It will be optimal after several cycles in automatic mode. It is not necessary to put the instruction SET WWRD 63 in the program.

**V – 6. Changing program automatically**

The automatic program change enables a program to be selected via an external dialogue without stopping the robot. Program 00 is reserved as a return address and as a switching point to the other programs requested.

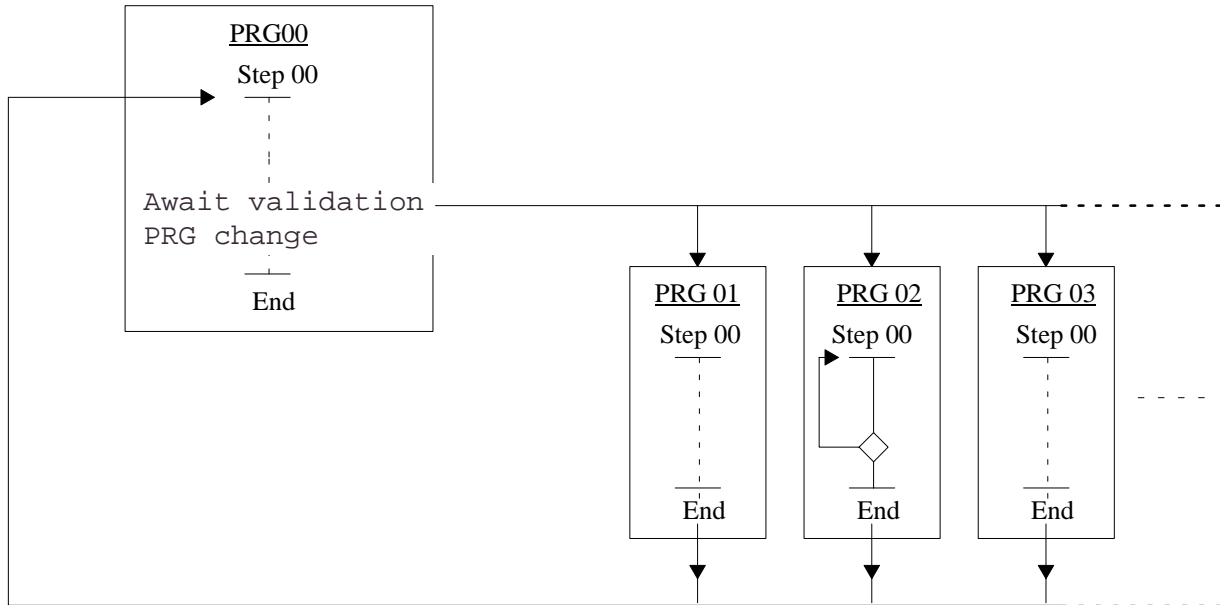


Figure 5 : Changing program automatically

**Principle :**

Program PRG 00 is reserved as a return address. It is also used as a switching point to the other programs. It must contain the instruction “Await validation PRG change“. See page 36 for how to enter this instruction.

**Operation :**

When the program PRG 00 is executed, the robot waits for the automatic program change validation (parameter 511 input or Bit 33). When this validation goes to 1, the robot executes the program whose number is coded, either :

- ▶ on the coding inputs (parameter 3 = 1 or 2),

or

- ▶ on WWRD 102 of the Euomap 17 dialogue (parameter 3 = 3),

or

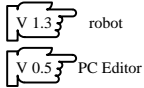
- ▶ in WRD 58 of the JBus dialogue (parameter 3 = 4).

Once the program has been executed, the robot returns to program PRG 00 and waits for the automatic program change to be validated again.

**Note :** If you want the program to run until a certain moment, you must program a loopback within the program. (PRG 02 in the example above).

**Home return :**


In the header of the return subroutines, it is possible to define the return to step 00 of program PRG 00. In this case, the distinction between simple and total has no influence on the stacking sequences.



## V – 7. Example of program with insert placing

For applications where an insert is placed in the mould, a special type of programming is necessary. This is because the IMM cycle must only be launched once the insert has been placed (i.e. once the mould has been accessed).

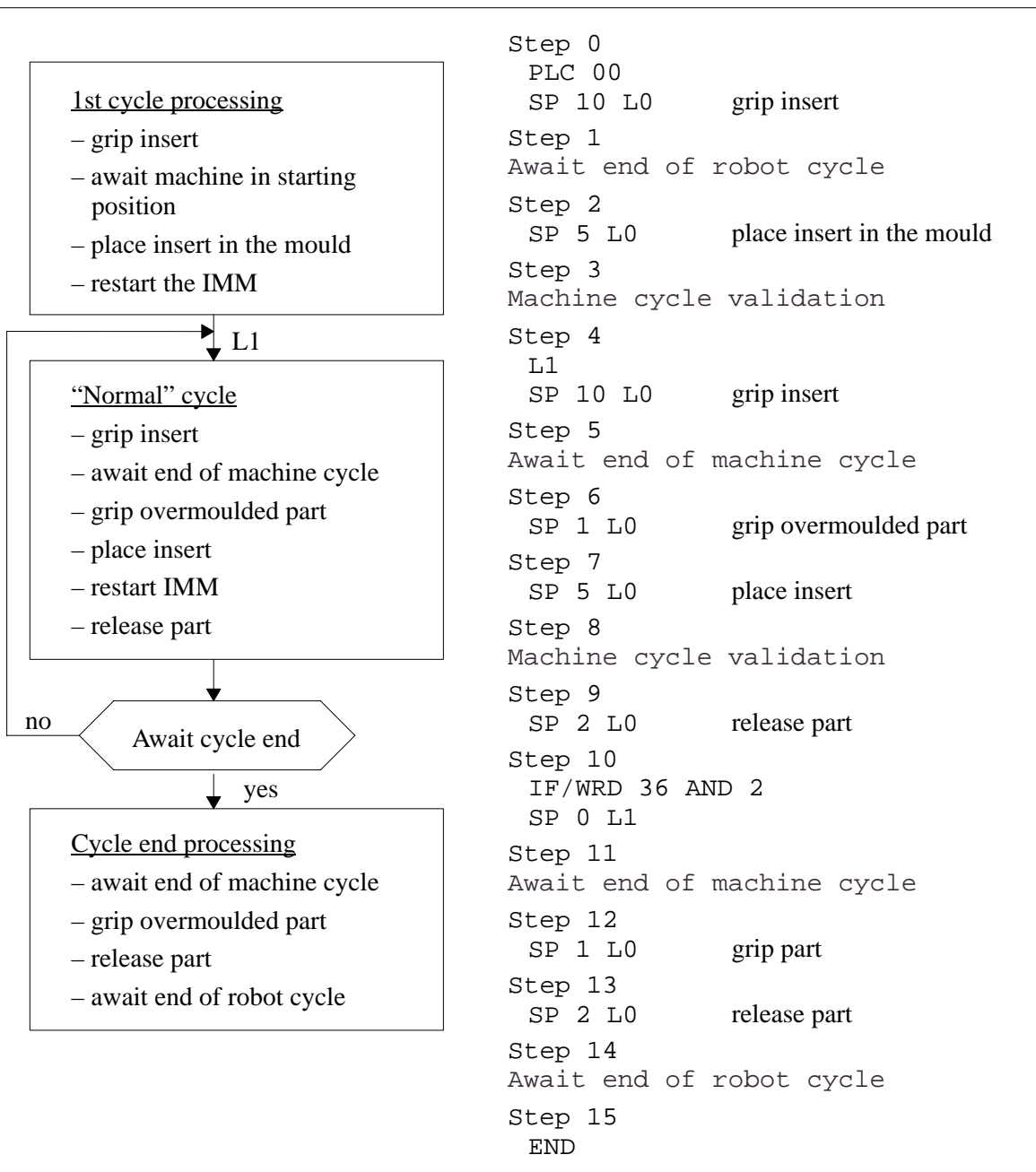
For the first cycle, the robot must first place an insert then launch the IMM cycle before it is able to descend into the mould to take an overmoulded part and place an insert.

This special type of program uses the instruction `Await end of robot cycle` which is accessible by pressing  twice.

Note : If this instruction is not accessible, change parameter 178 to 1 (see the “Configuration” manual).



This instruction does not wait for the mould to open before validating continuation to the next step. This test must therefore be programmed.





## VI – MEMORY MANAGEMENT

The robot is equipped with 2 internal and 3 external memory areas (optional). The figure below shows the possible contents of each of these 5 areas.

| Memory                                                   | Flashprom                                                                             | Standard module                                                                          | Module SAP                                                  | Diskette                                                                                |
|----------------------------------------------------------|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| PRG<br>SAP PRG<br>PLC<br><br>32 Kb or<br>128 Kb (option) | PRG<br>SAP PRG<br>PLC<br>SAP messages<br>Fault<br>messages<br>Parameters<br><br>64 Kb | PRG<br><br>PLC<br><br>16 Kb or 64 Kb                                                     | PRG<br>SAP PRG<br>PLC<br>SAP messages<br><br>16 Kb or 64 Kb | PRG<br>PRG SAP<br>PLC<br>SAP messages<br>Fault<br>messages<br>Parameters<br><br>1.44 Mb |
| <b>Internal memory</b>                                   |                                                                                       | <b>External memory</b>                                                                   |                                                             |                                                                                         |
|                                                          |                                                                                       | <div style="border: 1px dashed black; display: inline-block; padding: 2px;">option</div> |                                                             |                                                                                         |

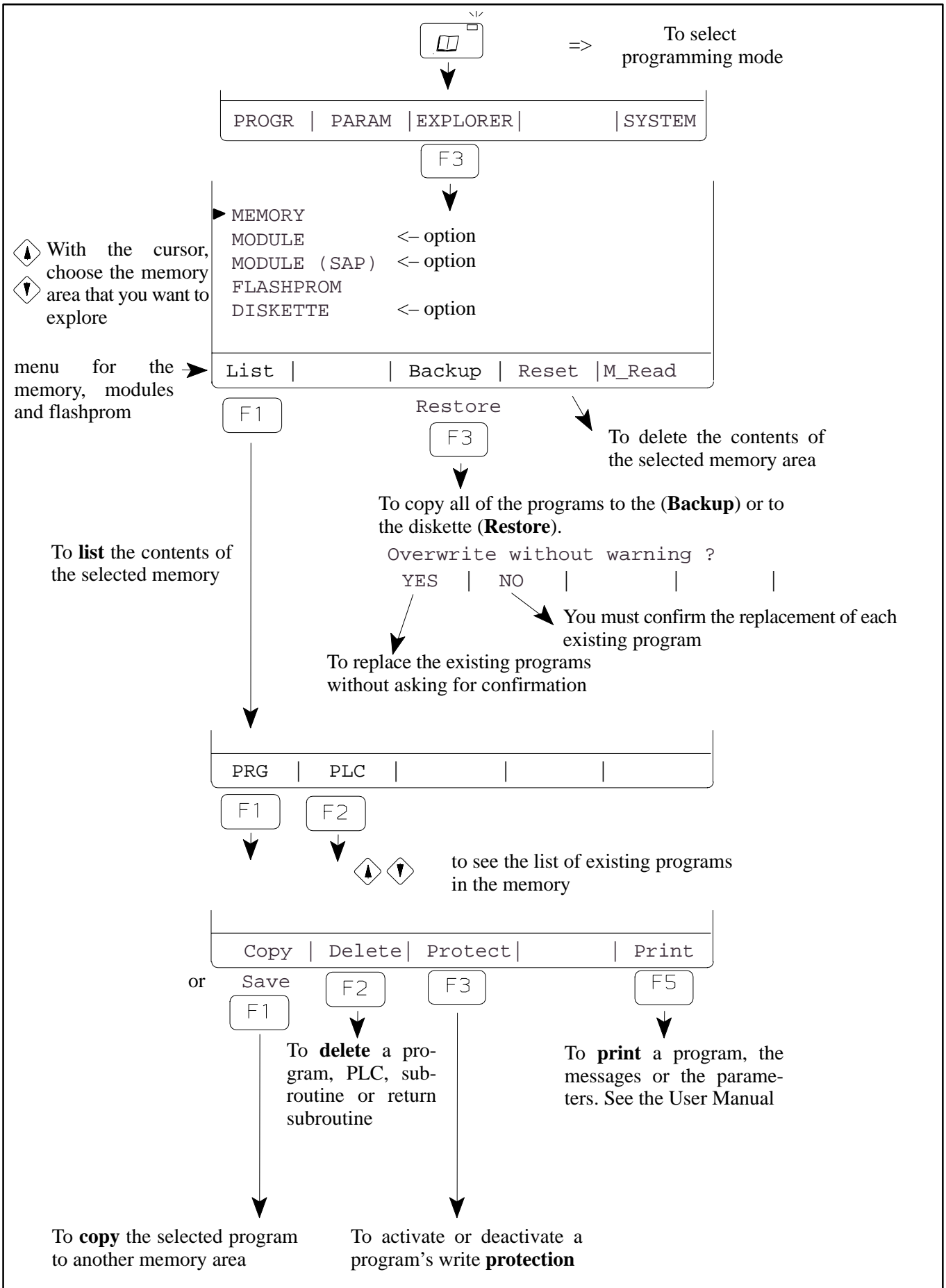
Figure 6 : Memory contents and type

Note : The programs can also be stored using the PC option. See the “AS900–II Editor for PC” software’s on–line Help.

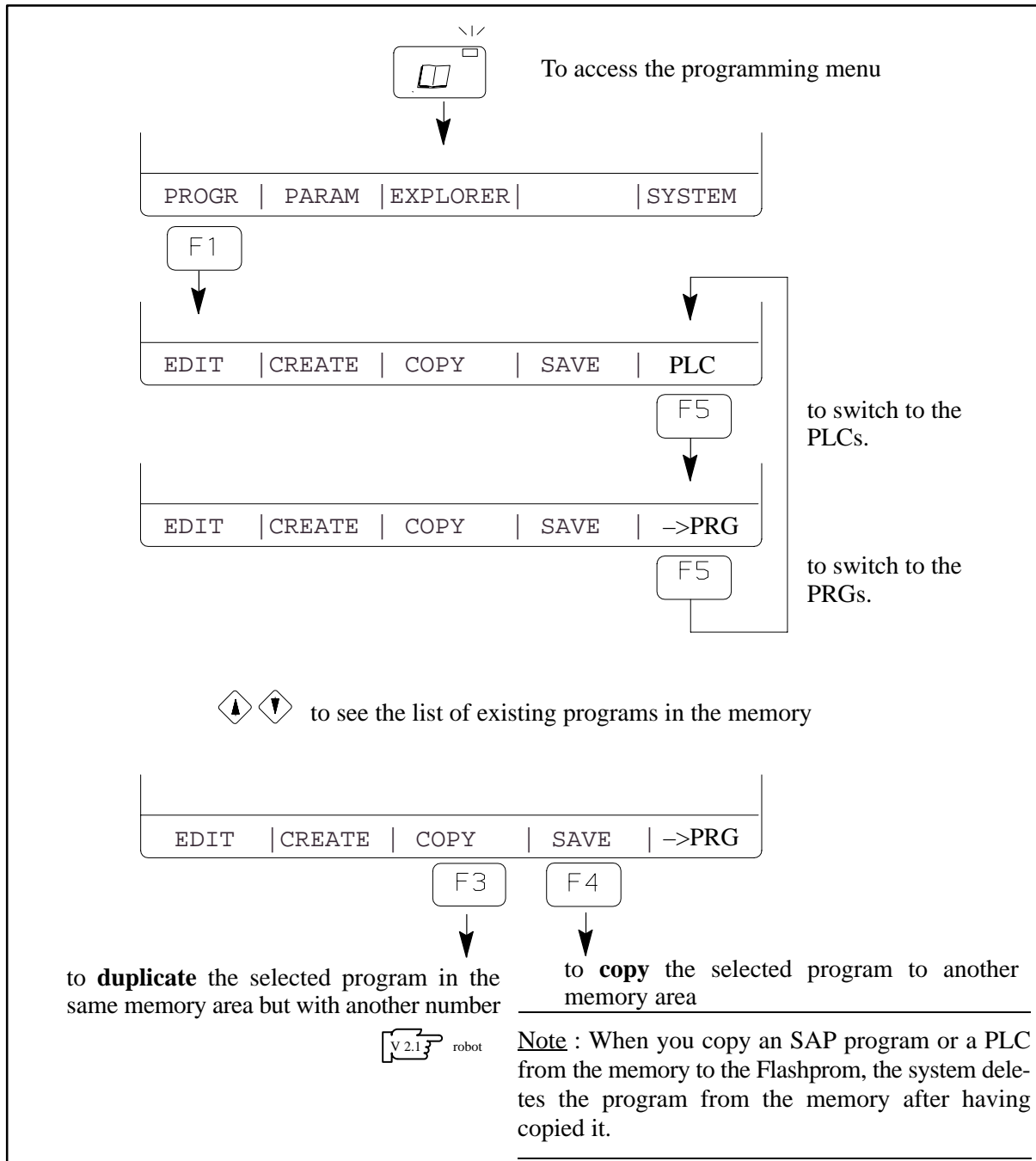
### VI – 1. The memory explorer functions

The user can consult the list of programs contained in each memory area. He can copy a program from one area to another, print it, delete it and protect it in write.

This protection is used to avoid deleting a program or overwriting it when copying from another memory area (not protected from modifications made from the PC).



**VI – 2. Memory management in programming mode**



Note : When you create a new subroutine, it is possible to copy the contents of an existing one. See page 15.

## VII – ANNEX

Print-out of the SAP message file corresponding to the example in figure 1 : page 1

```
I 01 Anticipation of Y advance
I 02 Anticipation of Z ascent
...
I 15
ACT 01 Reserved
ACT 02 Sprue-picker arm up
ACT 03 Reserved
ACT 04 Sprue-picker arm down
ACT 05 Reserved
ACT 06 Reserved
ACT 07 Sprue-picker arm forward
ACT 08 Sprue-picker arm back
ACT 09 Gripper horizontal
ACT 10 Gripper vertical
ACT 11 Grip part 1
ACT 12 Release part 1
ACT 13 Rotation 2 + direction
ACT 14 Rotation 2 - direction
ACT 15 Reserved
ACT 16 Rotation 2 intermed. position
ACT 17
ACT 18
ACT 19 Grip part 2
ACT 20 Release part 2
ACT 21 Grip part 3
ACT 22 Release part 3
ACT 23 Grip part 4
ACT 24 Release part 4
ACT 25 Grip part 5
ACT 26 Release part 5
ACT 27 Grip part 6
ACT 28 Release part 6
ACT 29 Grip part 7
ACT 30 Release part 7
ACT 31 Grip part 8
ACT 32 Release part 8
V 01 Z speed for descent in mould
V 02 Y speed for part grip
V 03 Y speed after part grip
V 04 Z speed for ascent in mould
V 05 X speed
V 06 Y speed outside mould
V 07 Z descent speed for release
V 08 Z ascent speed after release
V 09
...
V 15
T 01 Time delay after part release
T 02 Belt comand time
T 03
...
T 15
P 01 Grip part in the mould
P 02
P 03 End of return after part grip
P 04 Await mould open
P 05 Gripper rrientation
P 06 Release
P 07
P 08 Tool Change
P 09
P 10 Arm 1 Up
P 11
...
P 40
```

## – FIGURES –

|                                                 |    |
|-------------------------------------------------|----|
| Figure 1 : Cycle movements .....                | 1  |
| Figure 2 : Return subroutine branches .....     | 28 |
| Figure 3 : Conveyor belt (overhead view) .....  | 54 |
| Figure 4 : Stacking or palletization .....      | 60 |
| Figure 5 : Changing program automatically ..... | 66 |
| Figure 6 : Memory contents and type .....       | 68 |

---

Conair has made the largest investment in customer support in the plastics industry. Our service experts are available to help with any problem you might have installing and operating your equipment. Your Conair sales representative also can help analyze the nature of your problem, assuring that it did not result from misapplication or improper use.

## WE'RE HERE TO HELP

To contact Customer Service personnel, call:



## HOW TO CONTACT CUSTOMER SERVICE

**From outside the United States, call: 814-437-6861**

You can commission Conair service personnel to provide on-site service by contacting the Customer Service Department. Standard rates include an on-site hourly rate, with a one-day minimum plus expenses.

### **If you do have a problem, please complete the following checklist before calling Conair:**

- Make sure you have all model, serial and parts list numbers for your particular equipment. Service personnel will need this information to assist you.
- Make sure power is supplied to the equipment.
- Make sure that all connectors and wires within and between loading control and related components have been installed correctly.
- Check the troubleshooting guide of this manual for a solution.
- Thoroughly examine the instruction manual(s) for associated equipment, especially controls. Each manual may have its own troubleshooting guide to help you.
- Check that the equipment has been operated as described in this manual.
- Check accompanying schematic drawings for information on special considerations.

## BEFORE YOU CALL ...

*Additional manuals and prints for your Conair equipment may be ordered through the Customer Service or Parts Departments for a nominal fee.*

---

## EQUIPMENT GUARANTEE

Conair guarantees the machinery and equipment on this order, for a period as defined in the quotation from date of shipment, against defects in material and workmanship under the normal use and service for which it was recommended (except for parts that are typically replaced after normal usage, such as filters, liner plates, etc.). Conair's guarantee is limited to replacing, at our option, the part or parts determined by us to be defective after examination. The customer assumes the cost of transportation of the part or parts to and from the factory.

## PERFORMANCE WARRANTY

Conair warrants that this equipment will perform at or above the ratings stated in specific quotations covering the equipment or as detailed in engineering specifications, provided the equipment is applied, installed, operated and maintained in the recommended manner as outlined in our quotation or specifications.

Should performance not meet warranted levels, Conair at its discretion will exercise one of the following options:

- Inspect the equipment and perform alterations or adjustments to satisfy performance claims. (Charges for such inspections and corrections will be waived unless failure to meet warranty is due to misapplication, improper installation, poor maintenance practices or improper operation.)
- Replace the original equipment with other Conair equipment that will meet original performance claims at no extra cost to the customer.
- Refund the invoiced cost to the customer. Credit is subject to prior notice by the customer at which time a Return Goods Authorization Number (RGA) will be issued by Conair's Service Department. Returned equipment must be well crated and in proper operating condition, including all parts. Returns must be prepaid.

Purchaser must notify Conair in writing of any claim and provide a customer receipt and other evidence that a claim is being made.

## WARRANTY LIMITATIONS

**Except for the Equipment Guarantee and Performance Warranty stated above, Conair disclaims all other warranties with respect to the equipment, express or implied, arising by operation of law, course of dealing, usage of trade or otherwise, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.**

## – INDEX –

### A

ABS, 16, 18, 42  
Absolute, 16, 18, 42  
ACC, 45  
Acceleration, 45  
Action, 32  
Allocation, 38  
Anticipated restart, 64  
  . auto–adaptative, 64, 65  
  . programmed delay, 64, 65  
Arm free safety, 64  
Axis  
  . freed, 43  
  . linear, 44  
  . rotary, 44  
Axis movement, 42

### B

Backup, 68  
Belt, 53, 54, 60  
BIT, 37, 38, 39  
Bit  
  . status, 56, 57  
  . system, 25, 27, 56  
Branch, 24, 28

### C

Cascading SP, 23  
Changing  
  . entered value, 20  
  . program automatically, 36, 66  
  . program title, 21  
  . step, 13, 19  
  . tool, 8, 29  
CNT, 37, 38, 59  
Column, 56  
Comments, 61  
Condition, 24, 40  
Control, 48  
Conveyor belt, 53, 54

### Copy

  . a program, 70  
  . subroutine, 15  
  . un programme, 68  
Core puller, 34  
Counter, 37, 38, 58, 59  
  . part, 58  
  . stacking, 27  
Creation, 9  
CTL, 43  
Cursor, 19  
Customized fault, 63  
Cycle, 1, 4  
Cycle end, 56, 67  
Cycle time, 30

### D

DEC, 38  
Decrement, 38  
Delete  
  . a program, 69  
  . a step, 20  
  . an instruction, 19  
Double word, 37, 44  
Duplicate  
  . a program, 70  
  . subroutine, 15

### E

Editor  
  . PC, 10, 61, 68  
  . robot, 9, 19  
Ejector, 34  
END, 14, 52  
End  
  . column, 56, 59  
  . layer, 56, 59  
  . pallet, 56, 59  
End of cycle stop, 58  
End of slow approach, 46  
Escape, 14, 18  
Explore, 68



## F

FAL, 46  
Fault, 25, 49, 61, 62, 63  
Fault messages, 62  
FREE, 43  
Freeing sequence, 26  
Freeing sequences, 7  
FUNC, 45

## H

Header, 56, 59  
Home Return, 7, 26, 41, 49  
  . simple, 27  
  . total, 27, 56, 57, 59

## I

IF, 7, 24, 40  
If, 7, 40  
IMP, 47  
Imprecision, 47  
IN, 38  
INC, 38  
Increment, 38  
Input, 38, 55  
Insert  
  . a step, 21  
  . an instruction, 20  
Instruction, 32

## J

Jump, 24

## L

Label  
  . L, 23, 51, 52  
  . R, 52  
Language, 4  
Layer, 56  
LINE, 50

List, 68

## M

Machine cycle validation, 34, 64, 65  
Maintaining an input, 55  
MASTER, 48  
Memory, 68  
Mode  
  . programming, 9  
  . without robot, 56, 57  
Movement  
  . control, 43  
  . freed, 43  
  . master, 48  
  . relative, 43

## N

Name  
  . program, 11  
  . subroutine, 15

## O

OPA, 30  
Operand, 44  
Operation, 38  
Optimalization, 30  
Organigram, 4  
OUT, 37, 39, 41  
Output, 37, 41

## P

Palettization, 59  
Partial opening reached, 30  
PCO, 29  
PLC, 29, 51, 53, 54  
Preparatory function, 45  
Program  
  . main, 4, 23  
  . part, 9, 23  
  . SAP source, 3, 10

## R

REL, 43  
Relative, 43  
Reset, 38, 39  
Restore, 68  
Return address, 23, 26, 51, 52  
Robot OFF, 57  
Row, 56  
RST, 38, 39

## S

SAP marker

- . imprecision, 3
- . point, 2
- . time delay, 3
- . velocity, 3

SAP message, 71  
SAP point, 2  
Save, . all the programs, 68  
SBD, 64  
Selecting an instruction, 12  
Sequence, 2  
SET, 39  
SLA, 46  
Slow approach, 46  
SP, 4, 51  
Speed, 45  
SPP, 25, 53, 54, 56  
SR, 7, 8, 26, 41, 49, 51  
Stacking, 43, 56, 59

STK, 43, 59, 60  
Subroutine, 4, 23, 51

- . home return, 7, 26, 51
- . parallel, 25, 53, 54, 56, 57
- . stacking, 25, 56, 59
- . standard, 15

## T

Teaching, 18, 44  
Test, 24, 40

- . bit, 38
- . input, 38

TIME, 41  
Time delay, 41  
Timer, 37  
Title

- . program, 11, 21
- . subroutine, 15

Tool change position, 8, 29  
Trigger, 43, 48

## V

VCM, 64, 65  
VEL, 45  
Velocity, 45

## W

WORD, 37, 39  
Word, 37  
WORD, 37, 39, 44