E I P **E-300**

English

UNIPROG+ version 1.02ND

General purpose programming language for E300-ND and E-300-CMP controllers

Version: **June 6th 2007**

E.I.P. SA
CH-1667 Enney
Tél. : 026 / 921 80 40
Fax : 026 /921 80 49

Table of Content:

List of Figures:

List of Tables:

1 Introduction

The E-300 Motion Controllers are aimed at the market segment: special machine-tools, handling equipment's and assembly automation. Intrigate motion control problems and elaborate sequencing can be solved easily at low cost.

- The E-300 controllers are available for 1 or 2 axes, with step-motor translators for 2 phases motors.

- EIP SA has a proprietary language, the "PINX-E" and a program development tool, "APEX", to create applications running on his controllers. But these elaborate instruments are not optimal for practical situations.

- The UNIPROG program is a powerful tool to write and to debug applications directly at the controller keyboard.

- The UNIPROG program itself is written in the PINX-E language and it is a simple matter to produce enhanced versions, for example by the addition of application specific instructions.

The aim of this manual is to allow the inexperienced user to master UNIPROG after a thorough reading. Some knowledges of step motor techniques is a pre-requisite to avoid a trial-and-error approach. The reader is urged to read the sections 2 and 3 before attempting to write programs or to use the utilities.

The section 4 gives a complete description of the operating mode, starting at the power-up.

2 Memory organization

2.1 The user's storage area

Using the UNIPROG utilities, the user is able to organize its storage area in the battery backuped live memory. The SAVE utility writes the entire user's area into the Flash memory.

Within the user's area, a fixed portion is reserved to the configuration parameters (see section 4.4). The remainder is available for programs and numerical data. The UNIPROG editor stores lines. An instruction or a numerical data are always stored as one line. 3000 lines are available; 100 files may be opened within the line set. The files are numbered 00 to 99.

3 Coordinate System and Controller Configuration

This section defines the coordinate conventions, provides information about the motion generators and helps calculate the scaling factors.

3.1 Home Position

The stepper axes requires a home position before starting any useful work. Two different situations are common:

The machine has its own coordinate system, such as a jig boring machine for example
The coordinate system is fixed by the operator anywhere in the travel, example a rotary division table.

In the first case, the slide must be fitted with a home switch for the automatic and precise determination of the initial coordinate system.

The searching of the home position can be done by the operator or by the initialization program.

UNIPROG accepts a home switch anywhere in the travel.

If the home switch is not located at the ultimate end of the travel, it must be closed on one side and open on the other side in order to allow an unambiguous decision within the controller, see Figure 3-1.

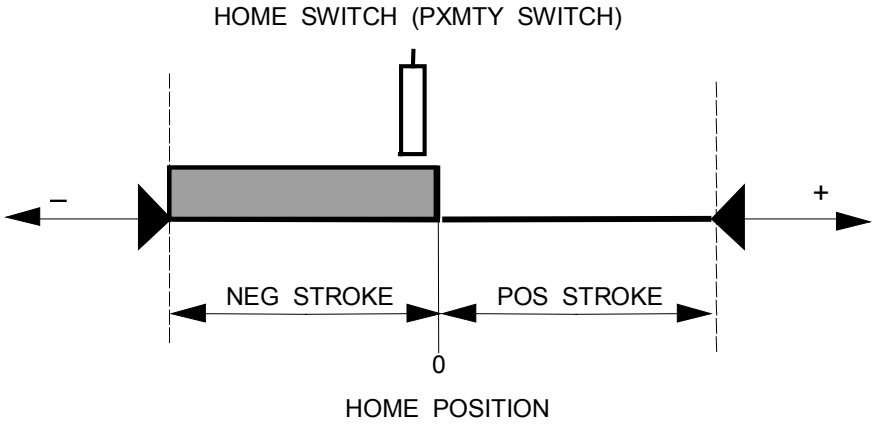


Figure 3-1 : Home and Travel Position

The process of finding the home position has three phases:

Phase 1:	The slide walks out of the home switch. (This phase takes place only if the home switch is active while entering the process.)
Phase 2:	Travel toward the home switch and stop with a ramp.
Phase 3:	Travel out of the home switch at reduced speed and immediately stops when the switch deactivates.

The phase 3 is responsible for the accuracy of the home position. It may be useful to notice the direction of the motion in phase 3 in order to take an eventual backlash into account.

The travel speeds are configuration parameters.

The E-300 motion controller has 4 inputs which can be used as Home -or Reference- switches: **INA0, INA1, INB0** and **INB1**.

If an axis does not use a home switch, the origin of the initial coordinate system will be fixed by the execution of home function without any motion.

The configuration menu has also provision for soft travel limits: STROKE+ and STROKE-.

The direction of the motions involved in the determination of the home position is governed by the sign of REF SPEED in the configuration.

If a soft travel limit is not convenient -with a rotary table, for example- STROKE+ and STROKE- must be set to 0. This situation requires NO REF for this axis.

3.2 Translation of the Coordinate System

Within UNIPROG, the travels are given either as "relative" or as "absolute" values. This applies to the programmed motions as well as to the JOGGING menu.

A relative motion is measured from the current axis position; the coordinate system is meaningless. For an absolute motion, the coordinate of the target point is given; the coordinate system actually used makes sense.

To create many coordinate systems, **tools** have been introduced (see Figure 3-2).

The Current Coordinate System is set-up by the instruction TOOL. The programmer may change the current system at will. If no TOOL instruction comes to execution, the current system is superimposed to the base system. Changing the current system may prove useful with multi-spindle machines or with handling equipment to translate from the "pick" to the "place" space.

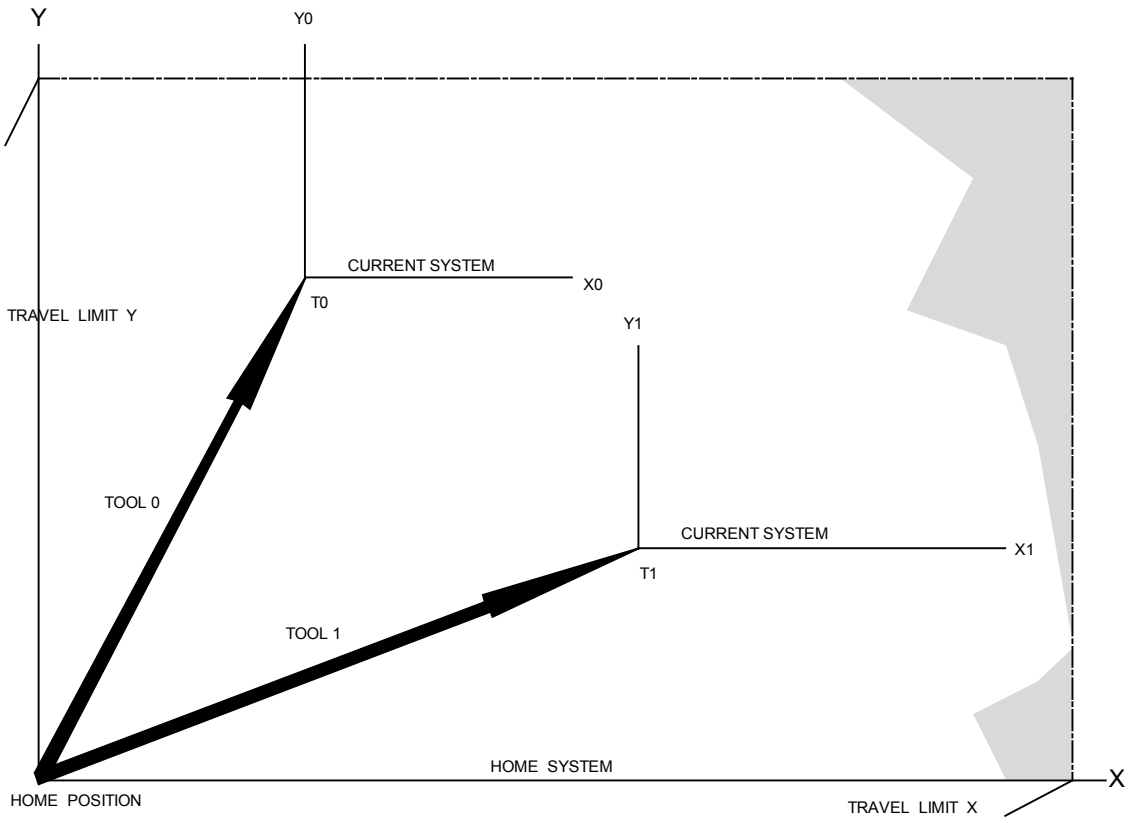


Figure 3-2 : The UNIPROG Coordinate System

3.3 Scale Factors and Constants

The factors and constants discussed in this section are to be given for each axis in the configuration menu.

3.3.1 The Length Scale Factor, SCALEK

This factor allows programming the travels directly in engineering units.

For a step motor drive, SCALEK is the number of pulses required at the input of the translator to effect one unit of travel.

The micro-stepping 2-phase translators needs 8 pulses for one full step. With the most usual step motors (1.8 degree/step), 1600 pulses produce exactly one revolution.

Examples:

Lead screw slide driven by a 1.8 degree stepper:
Timing belt 1:2 from motor to screw, thread pitch 5 mm, length unit 1 mm.

1600 pulses for one motor revolution,
3200 pulses for one lead screw revolution,
3200/5 pulses for 1 mm, then SCALEK = 640

3.3.2 Frequency Division Ratio, DIV

The Figure is a plot of the pulse frequency generated by the motion generator, i.e. the speed of the axis, during a single motion. The magnitude of the acceleration and of the deceleration decreases linearly with the speed in order to compensate for the weakening of the motor torque. The maximum of the frequency has to be set for each axis to preserve a sufficient torque margin at high speed.

The DIV parameter sets the highest frequency according to

DIV = 15'875/fmax [kHz]

The actual speed during a move is limited by the programmed speed. If the motor has a high torque at high speed, it may be advantageous to set fmax very high to obtain almost straight ramps.

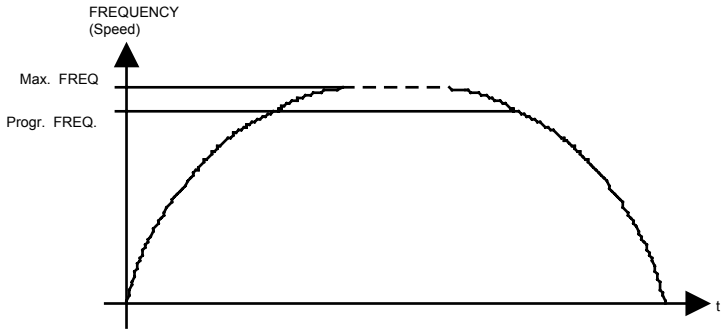


Figure 3-3 : Frequency or Speed versus Time

3.3.3 Acceleration Constant KUP, Deceleration KDN

These two parameters define the initial -or final- slope of the ramps. Their values are given in kHz/s or kpulse/s^2. As before, the frequency is the step frequency.

Values from 200 to 10000 kpulse/s^2 are generally used.

3.3.4 The Speed Scale Factor, FEEDK

The speed scale factor -or Feed Constant- allows the feed values to be given in engineering units: mm/s, m/min, rev/sec, etc.

For stepper drives:

FEEDK = pulse frequency for one feed unit [kHz]

Examples:

(For the electro-mechanical arrangements, see the examples of sect. 3.3.1)

a) The feed rate is expressed in en m/min.

SCALEK =640, i.e. 640 pulses for 1 mm
or 640'000 pulses for 1 m
or a frequency of 640 kHz for 1 m/s

640/60 kHz for 1 m/min

FEEDK = 10,667

b) Rotational in degree per second.

SCALEK = 111.111, i.e. 111.111 pulses for 1 deg,
or 0.111111 kHz for 1 deg/s

FEEDK = 0.111111

c) Speed in yard/min.

1 yard = 36 inches, 9676.19 * 36 for one yard
for 1 yard/min the required frequency is (9.6762 * 36)/60 kHz

FEEDK = 5.806

The actual feed rate can be at most equal to the maximum speed as set by DIV. The highest attainable speed, expressed in the chosen unit, is given by

15'875/DIV/FEEDK

N.B. The feed rate is in accordance with the programmed value only if the panel potentiometer is turned fully CW.

3.3.5 Soft Travel Limits, STROKE + / STROKE -

The soft travel limits must be expressed in the length unit of section 3.3.1. Enter the negative limit with a minus sign. Positioning and jogging motions are automatically limited to STROKE+, resp. STROKE- ; contouring motions are not a priori limited, but an over travel results in a fault situation, see section 7.3. 'Fault Processing'.

Due to internal register capacity limitations, the travel limit parameters have to meet:

STROKE+/- * SCALEK * DIV $< 2^{31}$ $2^{31} = 2.147 * 10^9$

Violation of the above rule is signaled while by the controller when the CONFIGURATION menu is left.

If soft travel limitations are not desired, STROKE + and STROKEN must have a value of 0.

3.3.6 Current Boost Command, BOOST

This parameter controls the action of the /BOOST line and its range is limited to 0..3.

BOOST = 0 :	/BOOST is always deactivated (high)
BOOST = 1 :	/BOOST is active during a move (low) and inactive whenever the axis is at rest
BOOST = 2 :	/BOOST is always activated (low)
BOOST = 3 :	/BOOST is high during a move and low at rest.

With the E-300 translators, BOOST is normally set to 1 ; at rest, the current is reduced to about 60% of it set value.

It is possible to set BOOST to 0 with small motors or to set BOOST to 2 if the full torque is required at rest.

4 Keyboard operating Mode and UNIPROG Utilities

This chapter describes the operation of the E-300 motion controller running under UNIPROG. The description starts at the power-up and supposes that all connections to the outer world are established. The menus are discussed in the sequence required for a first approach of the controller. The programming, the editor, the debugging are subject of the following chapters.

4.1 Switching the Power On.

The display shows the version of the system programs:

```
      UNIPROG+ V x.xx
E-300  x.xx          mm.dd.yy
```

After this, the E-300 may display:

```
RAM ERROR
FORMAT               YES  NO
```

Answer NO after the loading phase in order to configure the parameter " LAST TOOL NB " and to avoid the loss of file " 0 ". By answering NO the parameter " LAST TOOL NB " will not be used for opening file "0"and the control of the RAM memory loss will not be updated. By answering YES file "0" is opened and its contents are cleared. The control of the RAM is updated.

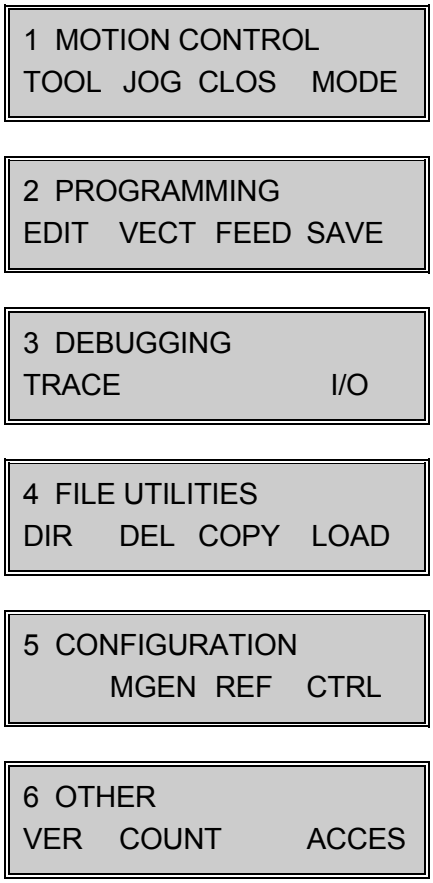
Menu #1:

```
1 MOTION CONTROL:
REF   JOG   CLOS   DISP
```

During the execution of any programs, the operator has access to all menus and it can make use of the utilities. Depressing the STOP button can stop a program.

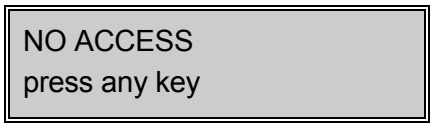
4.2 Menu Selection

The arrow keys are used to select a menu (↑ or ↓)



A menu offers up to 4 options; an option is entered by one of the function key, F1..F4. The lower line of the display contains the labels of the function keys. The ESC key is always active to escape from a sub-menu.

If the "No Access" message is displayed when attempting to enter a function, the access to this function is not granted, see section 4.3.1.



4.3 Menu "OTHER"

4.3.1 Access Flags and Access Code

In order to grant selective access, individual access flags can be assigned to the functions. For example, the machine operator may have access to the Jogging menu but not to the editor.
Whatever the status of the access flags, entering the access code grants the general access. After switching the power on, there is no access to functions with the flag set.

To have a general access, proceed as follows:

- Select the ACCES menu, press ENTER
- The message "ENTER ACCESS CODE" prompts the operator to enter the code

31415

- Press ENTER to terminate the entry.
- Press ESC to return to the menu selection.

Generally the entry of any number terminates with ENTER and typing errors can be corrected with CLR.

Entering the access code while the general access is granted will protect all functions with an access flag set.

To set the individual access flags, select the ACCESS menu and enter the code as described above. The functions -or group of functions- may be selected with the arrow keys. Entering a "1" gives the access, with a "0", the function is accessible only after introduction of the code.

4.3.2 Version Number

During the depression of the F1 key, the version numbers of the programs installed in the controller are displayed.

These data may prove valuable for service purpose.

4.4 "CONFIGURATION" MENU

4.4.1 MGEN, Configuration of the Motion Generators

The F2 key enters this sub-menu, starting from the CONFIGURATION menu. The parameters to be specified to the controller are organized in rectangular array: vertically, the arrow keys select the physical parameter; horizontally, the axis keys (X, Y) specify the axis to which a parameter belongs.

All these parameters have been discussed at chapter 3, a listing appears below.

DIV	Frequency divider
KUP	Acceleration Constant
KDN	Deceleration Constant
SCALEK	Length Scale Factor
FEEDK	Speed Scale Factor
STROKE +	Stroke in Positive Direction
STROKE -	Stroke in Negative Direction
BOOST	Current Booster Action, see 3.3.6

4.4.2 REF, Configuration of the Home (or Reference) Position

Enter the REF sub-menu with the F3 key. As before, the parameters are organized in a rectangular array.

REF INPUT	Number of the input used as reference input (Table 4-1 : Ref inputs). Specifying number 8 can inhibit the reference procedure. In this case, when reference function is involved, the position counter is reset to zero without any motion.
SPEED TO REF	Speed of the axis while searching its home position. Enter the speed in engineering units, as defined by FEEDK. A minus sign changes the direction of the home search.
CLOSURE GAP	see section 4.5.3, enter engineering units
SWITCH:	Enter "1" for a normally open home switch, "0" for a normally closed switch.
REF SPEED BACK	This entry is the ratio of the speeds of the two phases of the home position determination. For example SPEED TO REF is 100 mm/s and the ratio is 5. The axis will move out of its home switch at 20 mm/s. A high entry enhance the accuracy of the home position

REF INPUT Parameter	Physical input	Remark
0..7	IN 0..7	
8	-	REF without movement
9	Fault signal from translator	
18	INA from E-600-18 for Yaskawa	For Yaskawa, and in case of common limit-switch and reference switch.
60	INA0	
61	INB0	
62	INA1	
63	INB1	

Table 4-1 : Ref inputs

4.4.3 CTRL, Assignment of the Control Inputs

Enter CTRL with the F4 key. This sub-menu is intended to assign a physical input to three program execution functions. This assigned inputs then work as if they were ored with the panel keys, see also section 5.3.

If an external control input is not required, assign input 64 to this particular function.

EXTERNAL START	Program Start, uses a normally open contact
EXTERNAL PAUSE	Program Hold, normally closed contact
EXTERNAL STOP	Program Abort, normally closed contact.

This sub-menu also contains other parameters:

DISPLAY FORMAT 1-6	The number of digits to be written at right of the decimal point:
2 HAND START	Assign an input (0..7) for two hand start, the first input is selected with

	EXTERNAL START parameter. If two hand start is not wanted, enter 8 as number.
LAST TOOL NUMBER	Select the number of tools for machine-tools use. It limits the number of tools used, so as not to unnecessarily fill up program memory. Enter 0 for no tool. Entering 1 means the use of two tools.
MAX RPM 10 Volts	Select the ratio between DAC volts and RPM of the spindle. Enter the RPM number when DAC output = 10 volts.
FEED CTRL BY ADC	If 0, the speeds of step motors are controlled by the ADC input, if 1, the ADC input controls the speeds, via the potentiometer, if equipped.
LAST DELAYED OUTPUTS	Specifies a number of outputs for which the reset to zero is delayed. Possible values are between 0 and 7, giving the total number of delayed outputs from 0 to 7 accordingly. By specifying 0, no output is delayed. With a value of 7, outputs 0, 1, 2, 3, 4, 5, 6 and 7 are delayed. The default delay value is 1 s, this value can be modified within the program (see instruction SET).
TOOL L/R INVERT	(Used in contouring). Makes it possible to restore the correct tool side in a path, depending of the axis orientation. By entering the value 0, the side is considered to be normal, the reference of the surface is positive towards the upper right quadrant. Enter 1 for the other case.
LANGUAGE	Specifies the language in which certain messages will be displayed: 0=English, 1=French, 2=German.

4.5 MOTION CONTROL Menu

This menu introduces 4 sub-menus intended to manually move the axes and to display their positions.

The functions TOOL (Tool setting), JOG (Jogging) and CLOS (Closure check) are not available while a program is running.

4.5.1 TOOL, Tool Setting

In this menu, the origins and the diameter of the tools are accessible. 2 origins and one diameter are associated with each tool to achieve correction of the tool trajectory. The values are stored in file number 0 which is automatically opened when the parameter 'LAST TOOL NUMBER' is set. In the case of a wrong manipulation, file 0 does not have a suitable size and the following message appears:

```
FILE 0 NOT FORMATTED
Press any key
```

In this case, the parameter 'LAST TOOL NUMBER' must be reconfigured.

4.5.2 JOG, Jogging Motions and Tool setting

In this sub-menu, the upper display line shows the selected axis, its position in the basis coordinate system if no tools are used, or the position relative to the current tool. The lower line displays the current tool and the value of the incremental move effected by arrow keys. Depressing one of the arrow keys produces a move, the length of which is the value shown as "INCREMENT". If the depression ends before the end of the move, the axis stops with a deceleration governed by KDN. A new depression produces again a complete increment.
The F8 (JOG MODE) key move the cursor up and down. If the cursor stays in the lower line, a value can be entered (through the numerical pad) in lieu of the "INCREMENT" selected by F3

and F4. If the cursor in the upper line, a destination coordinate can be entered. Upon depression of the ENTER key, the move starts. The ENTER key must be held for the all length of the move.

This utility allows, in addition to its function of the manual displacement, to record the origins for the tools. The selection of the tool number is done by using F1 and F2 keys.

Recording a position is carried out in the following way:

1. Reach the position by jogging.
2. Select tool number (F1 and F2) and the relevant axis.
3. Press F5 key to authorize the input of the value of the new origin.
4. Enter the value on the numerical keypad, then validate it with ENTER key.

The new origins are stored in file 0 which contains 'FDATA'. To easily consult these origins, go to the TOOL menu accessible from MOTION CONTROL menu. Modifications can also be done from this menu.

Note:
For small adjustments, the correction of the origin can be entered in an incremental way using the arrow keys. the value of the increment will be added or removed from the origin. This is done without generating any displacement.

In the lower left quadrant the display indicates 'ALL'. If F1 is depressed, the correction of the origin will be applied to all of the tools. The selection is confirmed by the blinking of the led F8 and by the text at the bottom left 'ADJ ALL'. In such cases each hit on the arrow keys shifts the origin of all the tools.

The F7 (REF) key starts the reference search. **After switching the power on, a move is not enabled if the home position has not been set. However, incremental jogging motions are enabled.**

4.5.3 CLOS: Closure Check

A step motor drive is an open loop arrangement and it sometime desirable to have a periodic check of the validity of the actual position of the axis. The closure check verifies that the vector polygon closes around the home position within a given gap, the CLOSURE GAP. The check is effected in the direction adopted in determining the home position in order to eliminate backlash errors. A closure outside of the gap stops the program and issues an error message. In any case, the axis is at his home position after a closure check.

Depress F3 to enter the sub-menu, select the axis and depress REF. The result of the check is displayed on the screen:

```
CLOSURE WITHIN GAP
press any key
```

or

```
POS ERROR   1.234
press any key
```

If the axis has no home switch, the closure check does not work and a result within the gap will always be displayed.

After the check, the axis is at his home position.

4.5.4 MODE: Mode Selection and Axis Position Display

The MODE sub-menu displays the position of the two axes in the current coordinate system. MODE can always be called, even during the execution of a program. The display format is 6 digits, fixed point. The number of digits at the right of the decimal point has been selected at section 4.4.3.

The execution modes are explained in chapter 7.

4.6 Menu 'PROGRAMMING'

This menu contains all the functions needed to write, execute and to save programmes. The UNIPROG editor will be the subject of chapter 6 after having introduced the instructions.

4.6.1 VECT: Program Execution (Vectors)

In this sub-menu, the operator selects two programs:

- the POWER-ON-PROGRAMME, which comes to execution just after switching the power on,
- the START PROGRAMME, i. e. the program which is started after each depression of the START button (or after each activation of the designated external "Start" input).

It is important to notice that the Power-on-Program is executed after a full stop of the controller through the STOP button (or the corresponding external input). If this initialization program is not wanted, simply assign the program number 100 to the Power-on-Program.

4.6.2 FEED: Selected Feed Rates

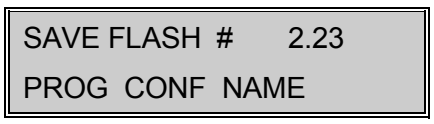
The velocity -or the Feed Rate- used as argument in the motion instructions are taken from the FEED table of this sub-menu. They are referred to by their position in the table, 0 to 6. The feed rates must be expressed in units selected while computing FEEDK.

4.6.3 SAVE: Saving User's Programs and Data in the Flash Memory.

This operation is not necessary after each modification of the files or the configuration, because the volatile memory (RAM), which is used during running and edition, is equipped with a battery. The saving gives a additional security for the datas in case of battery failure.

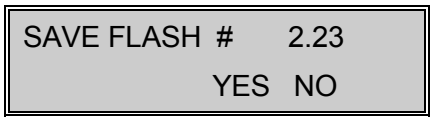
Three items can be saved separately:

1. All the files (programs)
2. Only the configuration
3. The number ('name') of the content.



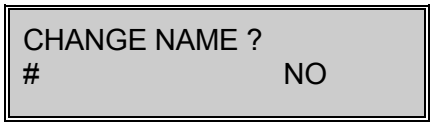
All open files are written to the Flash at once. The time to write the data strongly depends upon the amount of data to be rewritten in the Flash and it may last for a few seconds.

In order to avoid writing over valuable data, the warning

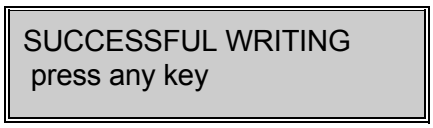


will be displayed. The identification code (here 2.23) is the actual code in the Flash. If the operator wants to write the data, it press F3 and the saving starts.

To change the code



Now, F4 aborts the saving. Entering a code -or NAME- will start the writting in the Flash after the depression of ENTER.



or

will be displayed.

A writing error is an indication of a Flash memory failure.

4.7 FILE UTILITIES, File Manipulation

The file utilities always act on the RAM contents.

4.7.1 DIR: File Directory

The screen gives information about all open files. A file can be opened by the editor or by the copy of an existing file.

```
FILE  SIZE  PROT  FREE
12     45    NO    670
```

The above example means:

- file 12 is open
- its size is 45 lines
- it is not protected
- there are still 670 lines free in the user area.

If the required file does not exist (was not opened), the message is as follows:

```
FILE   SIZE   PROT   FREE
18   NOT FOUND   670
```

The DIR utility may be used in several ways:

- To view all the files
- Use the arrow keys to explore the directory
- To view the status of a particular file

Enter its file number (followed by ENTER). One of the above screen shows the file status:

- To alter the protection status of a file
- The F3 key toggles the protect bit (YES = protected, NO = access granted)

A protected file cannot be edited or deleted. To give the end user a selective access to a subset of files, open the editor but close the DIR utility.

4.7.2 DEL: Delete a File

The screen prompts to enter the file number. In order to avoid unwanted deletion, the message "CLR to DELETE" prompts for a second key depression. CLR deletes the file, ESC returns to the menu "FILE UTILITIES" without deletion. An attempt to delete a protected file introduces the Directory screen with the file status displayed.

4.7.3 COPY: File Copy

The screen prompts to enter the SOURCE FILE number and then the DESTINATION FILE number. Several action can take place:

- The source file is not open: no action, return to "FILE UTILITIES"
- The destination file is not open: a new file is created
- The destination file is already open: The destination file and the source file are concatenated.

- The memory space available is too small for the file to be copied: no action done, only the screen warns the operator

```
TOO LARGE
 press any key
```

4.7.4 LOAD: Load the Flash Memory into the RAM

This operation is the reverse of the SAVE function, with two choice:

1. Restoring all the files
2. Restoring the configuration.

The LOAD destroys the RAM contents. Thus, a warning message is issued.

```
LOAD FLASH  #  1.03  ?
                    YES   NO
```

The identification code (1.03 in the example) is read out of the Flash. F4 returns to the base menu, F3 starts the loading.

```
SUCCESSFUL LOADING
 press any key
```

or

```
FLASH NOT FORMATTED
 press any key
```

The last screen indicates a wrongly formatted Flash. Format is done during SAVE operation.

4.8 "DEBUGGING" Utility

4.8.1 "TRACE" Utility

This utility makes sense during the execution of a program only. It shows the instruction actually being executed. As UNIPROG has a multi-task executive, the task to be traced has to be selected by F1. (F1 rotates the user's task number). The upper line of the screen shows the instruction in the editor format. The lower line displays the task number, the line and the program being traced, for example:

```
S: 1   L: 45   P: 12
```

means that the instruction being executed is in the simultaneous task 1 at line 45 of the program 12.

4.8.2 "I/O" Control Utility

This utility is intended to test and debug the hardware functions. The status of all UNIPROG controlled inputs and outputs are made visible; the state of the outputs can be set by key depressions. The output of the digital-to-analogue converter is also under control. For the meaning and the numbering of the I/O, please refer to the table in section 5.3.

The screen of the I/O Control menu displays one input, one output and the DAC value.

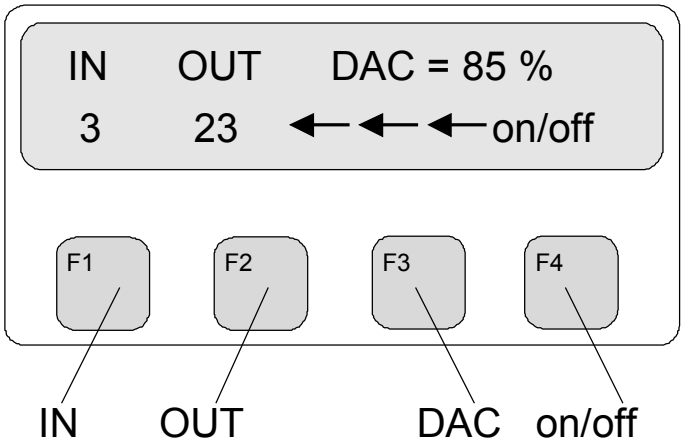


Figure 4-1 : I/O Utility

In the example above, the status of the input 3 is represented by the LED F10, the status of the output 23 by the LED F9 and the DAC output is 85 % of its end-scale value (i.e 8.5 V with the jumper in position a on the board.)
The F1, F2 and F3 keys move the cursor to IN, OUT and DAC. To select an input or an output, place the cursor at the desired item and enter the number or the value. The F4 key toggles the status of the selected output.

Notice, that this utility can be used while a program is running; this feature is a very practical way to fix hardware or software fault, such as a missing acknowledge signal.

According to Table 5-1, the pseudo-I/O 8, 9, 10 are activation status of the simultaneous tasks; the F4 key has no action on this items. The pseudo-I/O 11 to 15 are general purpose flags and their IN and OUT status are equal.

5 UNIPROG Instructions

The UNIPROG instructions are described using the mnemonics of the UNIPROG editor. The numerical code needed to enter the instruction at the keypad is given in the description. In the formal presentation below, the mnemonics are written in capitals, the arguments in lower case characters.

Instructions and pseudo-instructions pertaining to the contouring functions are left to chapter 8.

An instruction or a numerical data occupies a line in the user's storage area. We shall call "Line Address" or "Address" the number obtained by the concatenation of the decimal line number and the decimal file number, the file number being written with two places.

Examples:

1245 is the address of line 12 in program 45,
102 is the address of line 1 in program 2,
6 is the address of line 0 in program 6,...

The symbol "LINE/PROG" in the editor is just another denomination for "address".

There are several ways an instruction fetches the numerical value of its main argument (the displacement value in motion instructions, the duration in timing instructions, ...):

- The immediate argument: the numerical value itself is stored in the instruction.
- The direct argument: the instruction contains the address of the line where the value is stored.
- The indirect argument: the instruction contains the address of a pointer, the pointer holds the address of the numerical value.

For practical examples, see the instructions POSA, POSAD, POSAI,...

UNIPROG has a multi-task executive; 3 simultaneous programs may be running. Each program has its own accumulator which serves as destination or source register in several instructions.

5.1 Positioning Instructions

Six positioning instructions are available, 3 of them deal with "absolute" positioning, the other effect "relative" motions. The argument of an absolute instruction is a coordinate value, the argument of a relative instruction is a displacement value.

We have to point to an essential feature of UNIPROG: while effecting a relative motion, UNIPROG computes the target position in the absolute coordinate system, thus, repeated relative motions do not lead to an accumulation of rounding errors.

5.1.1 Absolute Positioning:

10	**POSA <*axis*> <*speed*> <*coordinate*> <*mode-e*>**
11	**POSAD <*axis*> <*speed*> <*address of the coordinate*> <*mode-e*>**
12	**POSAI <*axis*> <*speed*> <*pointer address*> <*mode-e*>**

5.1.2 Relative Positioning:

14 POSR <*axis*> <*speed*> <*displacement*> <*mode-e*>
15 POSRD <*axis*> <*speed*> <*address of displacement*> <*mode-e*>
16 POSRI <*axis*> <*speed*> <*pointer address*> <*mode-e*>

"speed" argument: Integers 0..7 are legal. From 0 to 6, the velocity is taken from the table (see § 4.6.2). If speed = 7, the velocity is supposed to be in the accumulator.

"Mode-e" argument:

Mode-e = 0:	the motion is accounted for but not yet executed
Mode-e = 1:	all accounted motions for the designated axis are executed
Mode-e = 2:	all accounted motions in all axes are executed
Mode-e = 3:	generates a straight vector in the selected space, see chapter 8.

The various execution modes allow the programmer to add relative motions and then to effect a single move. Of course, this does not work for absolute motions, as only the last entered coordinate is significant.

5.1.3 Tool setting

19 TOOL <*tool number*>

The instruction “TOOL” sets a new reference valid for all subsequent positioning instructions. The components of the translation vector of the origin are stored in the file " 0 " (see § 4.5.2). The " TOOL" table allows consultation or modification of these origins. These origins can also be memorized and validated directly from within the jogging.
The parameter "tool number" selects the group of 3 components associated with a tool number.

5.2 Other Motion Instructions

5.2.1 Home Position Search:

17 REF <*axis*>

The home -or reference- position search is done as explained in section 3.1. The velocity is a parameter in the configuration, see § 4.4.2.

5.2.2 Closure Check:

18 CLOS <*axis*> <*speed*>

The closure check function was explained in § 4.5.3. The "speed" argument has the same meaning as in the positioning instructions.

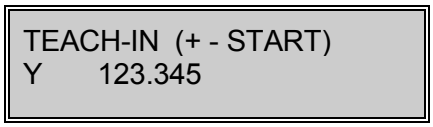
5.2.3 Teach-In Instruction:

13 TEACH <*axis*> <*speed*> <*address*>

The arguments "axis" and "speed" have been discussed under Positioning Instructions.

The TEACH instruction makes it possible to enter -or to modify- a position by teaching. The instruction comes to execution in the step-by-step mode only, see section 7.1. It is mandatory to set the pause flag as explained in section 6.4.

When the program stops at the TEACH instruction, the screen shows:



The lower line displays the axis and the contents of the line "address". There are two possibilities to modify this contents:

- an effective motion of the axis by the JOG keys,
- entering corrections at the key pad.

Both methods can be used in the same teach-in session. The corrections are always added to the contents of the line "address". This line can be the immediate, the direct or the indirect argument of a positioning instruction or a data line.

With the JOG keys and the potentiometer, the position can be accurately taught. Several motions are permitted. To resume the program execution, press START.

5.2.4 Setting of variables:

83 SET <*parameter number*> <*value of the parameter*>

Parameter number:

0	PASSE	value of the pass (PECK instruction).
1	GAP	(PECK instruction)
2	DELAY	'bottom of drilling' delay (PECK instruction)
3	BRK-D	delay before switching off the outputs (LAST DELAYED OUTPUT)

5.2.5 Peck cycle (drilling):

84 PECK <*axis*> <*slow speed*> <*'bottom of drilling' position*> <*mode-d*>

The instruction determines the number of passes. The 'bottom of drilling' position is absolute in the selected tool area.

Four modes are possible:

Mode-d = 0:	Pecking with return to the start position, at the end of the cycle.
Mode-d = 1:	Chip-breaker with return to the start position, at the end of the cycle.
Mode-d = 2:	Pecking without return.
Mode-d = 3:	Chip-breaker without return.

The fast approach stops just before the material. This gap is set by to default to 0.1 mm and can be changed with the instruction SET. A delay is observed at the bottom of the drilling in modes 0 and 1. Its default value is 0.1 s, it can be changed with the instruction SET. The modes 2 and 3 can be used to chain different drillings to have a progression.

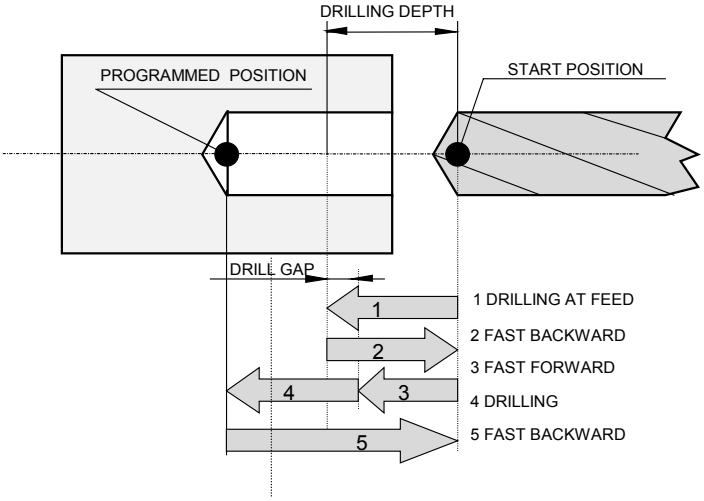


Figure 5-1 : Peck Cycle

5.2.6 Tapping instruction:

81 TPING <*axis*> <*pitch*> <*final tapping position*>

This instruction determines the feed of the axis, in regard of spindle speed and pitch of thread. The instruction SPVEL determines the spindle rotation speed controlled by a frequency converter. The pitch must be given in the FEED table.

At the end of the tapping, the output number 7 is inverted, therefore it can be wired on the input 'direction' of the frequency converter. The spindle rotation is reversed when the final position is reached. The spindle rotation is not measured and therefore does not affect the duration of the tapping. Consequently, the depth of the tapping can change in regard of the torque.

Example:

```
00    POSA     X 4  0.00  1
01    SPVEL    600              ; Spindle rotation speed
02    ON       7                ; Spindle rotation
03    ON       6                ; Start the spindle
04    TPING    X 3  50          ; Start the tapping
```

Important note:
The feed is determined without a measure of the spindle speed, therefore a special tapping tool with length compensation MUST be used.

5.2.7 Rectilinear Displacement

46 CORR <*speed*>

The instruction CORR creates a rectilinear displacement and at a speed which is indicated at the starting position of the contour by taking into account the radius of the tool and its direction.

5.2.8 Angle

86 ANGLE <*speed*> <*value*> <*mode-e*>

The instruction ANGLE gives the angle in relation to the 0 fixed at three o'clock. A shift of the 0 is possible with the instruction ORGA. The positive direction is counter-clockwise. <Value> is given in degrees.

mode-e = 0:	the angle is recorded while waiting for the execution of the movement with the instruction RADIUS.
Mode-e =1 or 2:	the displacement on each axis is created each one at their respective speed.
Mode-e = 3:	the displacement is linear up to the next point.

5.2.9 Radius

87 RADIUS <*speed*> <*value*> <*mode-e*>

The instruction RADIUS gives the radius in relation to the 0 fixed in relation to the center. The radius is always positive.

Mode-e = 0:	the radius is recorded while waiting for the execution of the movement with the instruction ANGLE.
Mode-e= 1 or 2:	the displacement on each axes is created each one at their respective speed,
Mode-e = 3:	the displacement is linear up to the next point.

5.2.10 Angular Shift

88 ORGA <*angular shift*>

If the zero fixed at three o'clock is not appropriate, an angular shift is possible toward a positive or negative direction.

5.2.11 Reference for automatic tool adjustment

89 ZTOOL <axis> <input number> <direction>

This instruction is useful to take automatic tool origin (in case of tool wear and tear for example).

This instruction allows memorizing in the accumulator the absolute position value on an axis when a chosen input is active. This position value can be then treated and memorized in another variable (for example FDATA from tool origin to modify).

To realize the tool origin movement this instruction uses speed parameters defined in the REF menu of the axis.

Input number can be 0 to 7.

Example:

We take an X-Y system, with automatic adjustment of Y tool origin during the cycle.

5 tools are defined:

0: machine origin
1: piece origin
2: origin for fast approach near the detector
3: current piece origin
4: detector origin

Tools 0, 1, 2 and 4 are adjusted only one time. Tool 3 is adjusted one time with the same value as Tool 1 and is then by program modified during the cycle.

File 1

```
 0 01  19   TOOL    3                          ; Working with tool 3:
 1 01  10   POSA    X     0   0.0000   0       ; (example)
 2 01  10   POSA    Y     0   0.0000   2
 3 01  19   TOOL    2                          ; Automatic adjustment:
 4 01  10   POSA    X     0   0.0000   0       ; detector approach (tool 2)
 5 01  10   POSA    Y     0   0.0000   2
 6 01  89   ZTOOL   Y     2   0                ; Measure, position in accumulator
 7 01  92   SUBD    2100                       ; calculation: Difference with original detector value (tool 4)
 8 01  91   ADDD    600                        ; Addition of correction and piece origin (tool 1)
 9 01  55   STORD   1600                       ; memorization in  current piece origin (tool 3)
10 01  60   JMP     1
```

5.3 Input/Output and Display Instructions

These instructions allow the programmer to wait for an input, to branch conditionally upon an input status and to set/reset an output.

5.3.1 Wait for an Input:

20 WAIT0 <*input*>
21 WAIT1 <*input*>

The program holds as long as the designated input is 0, resp. 1.

5.3.2 Conditional Branch:

22 BRINO <*input*> <*address*>
23 BRIN1 <*input*> <*address*>

The program execution is transferred at "address" if the designated input is 0, resp. 1. Otherwise, the program executes the next instruction. About the branch address, see the note in section 5.6.

5.3.3 Output Control:

28 OFF <*output*>

The designated output is set to 0 (OFF) or to 1 (ON).

Input	Item	Output	Item
0	IN(0)	0	OUT(0)
1	IN(1)	1	OUT(1)
2	IN(2)	2	OUT(2)
3	IN(3)	3	OUT(3)
4	IN(4)	4	OUT(4)
5	IN(5)	5	OUT(5)
6	IN(6)	6	OUT(6)
7	IN(7)	7	OUT(7)
8	SIM(0)	8	SIM(0)
9	SIM(1)	9	SIM(1)
10	SIM(2)	10	SIM(2)
11	FLAG(0)	11	FLAG(0)
12	FLAG(1)	12	FLAG(1)
13	FLAG(2)	13	FLAG(2)
14	FLAG(3)	14	FLAG(3)
15	FLAG(4)	15	FLAG(4)
16..49	IN(16)..IN(49)	16..63	OUT(16)..OUT(63)
50..59	KEY("0".."9")		
60	INA0		
61	INB0		
62	INA1		
63	INB1		

Table 5-1 : Inputs and Outputs

The inputs IN(0..7) are dedicated as home switches, but they are general purpose in nature, thus, they can be tested by the WAIT and BRIN instructions.
The pseudo-I/O SIM0, SIM1, SIM2 are the activation status of the simultaneous UNIPROG tasks.

FLAG(1..5) are general purpose flags, which can be set/reset by ON/OFF and tested by WAIT and BRIN.

IN(50..59) are the state of the numerical keys 0 to 9.

IN(16..49) and OUT(16..63) are implemented by the I/O extension modules. Each module must be given an address by the switch setting of Table 5-2.
One input and one output module may have the same address.

O = Open,
C = Closed

Switch Setting 4 3 2 1	Address IN OUT
O O C O	16...23
O O C C	24...31
O C O O	32...39
O C O C	40...47
O C C O	48...55
O C C C	56...63

Table 5-2 : I/O Module addresses

OUT(0) à OUT(7) are outputs implemented within the basic E-300 housing. They are available through the back panel connector.

5.3.4 Complement of output:

95 CPL <*output number*>

This instruction inverts the output state.

5.3.5 Display of a value

79 DISPD <*position*><*pointer address*>

<*position*> is a number between 0 and 3 which gives the area where to display the value:

0 = Upper left corner

1 = Upper right corner

2 = lower left corner

3 = lower right corner

<*pointer address*> contains the address where is the value to be displayed. (IDATA or FDATA)

80 RBW

This instruction restore the window which were displayed before DISPD.

5.4 Number Handling

UNIPROG works either with real -or floating- numbers and with integers. Velocities, positions, displacements, times are floating numbers while addresses and a number of cycles are integers.

With a few exceptions, the arithmetic and data handling instructions work with real numbers. A line holding an instruction with an immediate argument can be the source or the destination of an instruction dealing with real numbers. Thus, the program is able to modify itself.

5.4.1 Load Accumulator

Load Accumulator Immediate:

50 FLOAD <*floating number*>

51 ILOAD <*integer*>

Load Accumulator Direct:

52 LOADD <*address*>

Load Accumulator Indirect:

53 LOADI <*pointer address*>

5.4.2 Store Accumulator

Store Accumulator Direct:

55 STORD <*address*>

Store Accumulator Indirect:

56 STORI <*pointer address*>

5.4.3 Pointer Incrementation/Decrementation :

58 INCD <*address*>

59 DECD <*address*>

These instructions are used with the indirect addressing: they move the pointer to the next (INCD) or to the previous (DECD) line. INCD adds 100 to the composite address contained in the pointer. The file part of the composite address remains unchanged.

5.4.4 Save a Variable into the Flash Memory:

54 SAVE <*address*>

The variable in line "address" is saved in the Flash memory at the same address. The write operation to the Flash memory may last for several tenths of a second, therefore this instruction is not recommended in portions of programs where timing is critical.

5.4.5 Load the Digital-to-Analogue Converter (DAC):

57 SPVEL <*rotations per minute*>

The parameter determines the speed of the unit connected to the DAC. The parameter 'MAX RPM 10 volts' in the configuration must be set (for example the value 100 permits to give the speed in percentages).

5.4.6 Frequency Converter control:

85 MOTOR <*motor number*> <*rotation speed*>

This instruction controls the spindle motors by ensuring necessary delays in order to change the frequency converter contactors. The frequency converter must absolutely be changed without current and therefore at a speed of zero. The MOTOR instruction cycle sets the speed to zero then waits the delay set in the SET 4 BRK-D instruction before reactivating the speed (in rotation per minute).

With motor number 0, all motors are stopped.
With motor number 1, motor number 2 is stopped and motor 1 starts.
With motor number 2, motor 1 stops, motor 2 starts.
etc... to motor number 7.

5.5 Program Control Instructions

5.5.1 Unconditional Jump:

60	**JMP <*address*>**

The program execution is unconditionally transferred at the line 'address'.

Important Notice: The UNIPROG editor has an insert/delete line function, and deletion alter the numbering of the lines in a file. To avoid line referencing problems, it is recommended, but not mandatory, to organize the program files in order to jump or call only at line 0.

5.5.2 Subroutine Call:

61	**CALL <*address*>**

The program execution is transferred at "address", the beginning of the subroutine. At the end of the subroutine, execution resumes at the main program at the line just after the Call. Up to 10 nesting levels of subroutines are allowed.

5.5.3 Program End, Subroutine End:

62	**END**

At the end of a main program, this instruction transfers the control to the operating system. At the end of a subroutine, END has the function of a Return. If the last instruction of a program or a subroutine is also the last line of the file, no END instruction is required.

5.5.4 Repeat Loop:

The instructions in this section are intended to built repeat loops without any overhead. Up to 10 loops can be nested and a loop can extend over several files. A loop begins at the REP or REPD instruction and ends with a ENDRP instruction. The argument of REP must be an integer (number of loops); with REPD, the contents of the designated line may be an integer or a real. In the later case, the decimal fraction is discarded.

63	**REP <*integer*>** immediate argument	
64	**REPD <*address*>**	direct argument, integer in "address"
65	**ENDRP**	end of the loop

5.5.5 Simultaneous Task Activation:

67	**SIM1 <*address*>**
68	**SIM2 <*address*>**

The simultaneous Task #1 or #2 starts execution at "address".

A simultaneous task is terminated when it encounters a END instruction or the end of the file. A simultaneous task can be paused by switching the control flag SIM1 (SIM2) off; the paused task resumes its operation when the SIM flag is again switched on. The SIM flags can be tested to gain information about the activity status of the tasks. Calling an already active task at another address transfers the control of the task to the new address. The STOP key abort all active tasks.

5.5.6 Conditional Branch on Accumulator Contents:

These instructions are intended to test the result of an arithmetic instruction. If the condition meets, the program control is transferred to "address", if the condition does not meet, the program continues in sequence. Refer to the notice in section 5.5.1.

24	**BRM <*address*>**	Branch if the contents of the accu. is negative
25	**BRP <*address*>**	Branch if the contents of the accu. is positive or zero

26	**BRZ <*address*>**	Branch if the contents of the accu. is zero
27	**BRNZ <*address*>**	Branch if the contents of the accu. is non zero

5.6 Timing Instructions

70	**WAIT <*time*>**	Immediate Argument
71	**WAITD <*address*>**	Direct Argument, time is in "address"

These instructions are dead timers. The time is given in seconds and the line addressed by WAITD must contain a real number.

5.7 Arithmetic Instructions

One operand is the contents of the accumulator, the other operand is the contents of the direct argument. The result returns to the accumulator.
If the direct argument is a line opened by an IDATA, integer operands are assumed. In all other cases, floating numbers are assumed.

91	**ADD <*address*>**	Accu = Accu + [address]
92	**SUBD <*address*>**	Accu = Accu - [address]
93	**MULD <*address*>**	Accu = Accu . [address]
94	**DIVD <*address*>**	Accu = Accu / [address]

5.8 NOP and Directives

90	**NOP**

The NOP (No Operation) instruction does nothing. It is useful to reserve space in a program for future modifications. While editing, a line not yet opened appears as a NOP.

98	**FDATA <floating number>**
99	**IDATA <integer>**

FDATA and IDATA are not true instructions but directives to declare numerical variables. The arithmetic instructions are directed by the declaration.

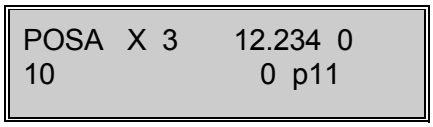
It is worthwhile to notice that the directives 98 and 99 act as NOP when written in a program. Thus, it possible to declare numerical variables anywhere in a program.

5.9 Pause Flag

A Pause Flag can be set at any instruction in a program. The pause flag has no action if the program runs in mode 1, but the program will pause at each flag in mode 2. For more details, see chapter 7.

6 The UNIPROG Editor

The editor is entered through the "PROGRAMMING" menu, as mentioned in section 4.6. The screen prompts the operator to enter the file number to be edited. The contents of the line 0 of the selected file comes to the screen.



The upper line displays the instruction mnemonics and the value of its arguments, if any. The lower line gives the numerical code of the instruction (just under the mnemonics) and the line and file number. The cursor is blinking on the instruction name.

If the selected file is not yet open, a NOP instruction appears and the file will be opened as a significant instruction is entered.

6.1 How to Read a Program ?

Arrow Keys:	'UP' goes to the previous line, 'DOWN' goes to the next line.
F2 Key:	Enter the line number to go to the line you want to examine
ESC Key:	Returns to the editor prompt menu; a new file can be selected. A second depression of ESC is needed to return to the base menu.

6.2 How to Modify the Contents of a Program Line ?

Writing a new line in a program amounts to modify a line containing a NOP. Thus, it is sufficient to know how to modify a line.

ENTER Key:	Moves the cursor to the next argument at right. The lower line informs the operator about the argument to be entered. After the last argument -or after the instruction itself if it does not require an argument- the whole line is stored and the next line is put on the screen.
CLR Key:	Moves the cursor to the argument at left. Useful to correct an erroneous entry.
F5 Key:	If the cursor stays at the instruction symbol or at a symbolic argument, the F5 key presents all possible choices in sequence (in increasing numerical order). F5 has no action when the cursor points to a numerical argument.

To enter an instruction, it is not necessary to use the F5 key to scan all possible entries. It is faster to directly enter the numerical code. With same practice, the codes are easily memorized, at least the class to which it belongs. For example: positioning instruction: class 10, data handling: class 50, timing: class 70,.. A few depressions of F5 will then get the desired symbol. Please notice that no ENTER key is needed to enter the numerical code of an instruction, the number is automatically entered after two figures. It is then necessary to use the CLR key to place the cursor.

It is always possible to examine (with the arrow keys) the other lines while in the process of modifying an instruction.

Example: Enter the POSR instruction into an empty line:

- POSR code is 11. Enter 11 (ENT key not required). The POSR symbol is displayed in the upper line.
- The display now prompts the operator to select the axis, the selection can be dona with F5 or directly by entering 0 for X, 1 for Y.
- ENTER introduces the next argument: the speed (SEL.SPEED). Enter a decimal number 0..7. F5 can't be used here. The next argument is the displacement (DISP'MENT) , which must entered in engineering units.
- The last argument is the execution mode (EXEC MODE), F5 is active, but the direct entry is faster: 0, no immediate execution, 1, the designated axis moves, 2, all instructed axes move, 3, vector generation. ENTER stores the line and put the next line to the screen.

To modify a single argument in an instruction, put the cursor on the argument and enter the new value. Then depress ENTER several times until the line is stored (next line displayed).

6.3 How to Insert and Delete a Line ?

The F3 key inserts a line at the displayed line number.

Example: The line 12 is at the screen and its contents is the instruction WAIT. After an insertion, the line 12 contains a NOP and the Wait instruction occupies the line 13.

The F4 key deletes the displayed line. The next line takes the number of the deleted line and comes to the screen.

6.4 How to set a Pause Flag ?

The Pause Flag is set or reset by the F1 key (toggle action). The LED of the F1 key displays the presence of the flag. The pause flag is effectively stored while storing the line to which it belongs. i.e. if the editor goes to the next line upon depression of ENTER (The arrow keys do not store a line !).

7 Programme Execution

The program execution is governed by the push buttons START and STOP, by the inputs designated in the CTRL configuration and by the execution mode selected in the menu MODE.

7.1 The Execution Modes, menu 'mode'

MOD1 (F2):
Normal execution mode. The Pause Flags in the instructions are ignored. The START button is lit when the program is running.

MOD2 (F3):
The Pause Flags stops the program before the execution of the flagged instruction. The simultaneous tasks go on unless a flagged instruction is encountered.

During the pause, the START LED is blinking. A depression of START restart the program till the next flagged instruction. This mode is especially useful with the TRACE utility.

SAT (F1):
In this mode, the displacement speed is limited, but the pause flags are treated as in mode 2.

7.2 START and STOP Key Functions

Remember that the inputs designated by the CTRL configuration are effectively ORed with this push buttons.

START
When the pilot lights in the START and STOP buttons are off, a depression of START effectively starts the program designated as "START PROGRAMME" by the VECT menu. If the red STOP light is on, the program designated as "POWER ON PROGRAMME" comes to execution.

PAUSE (MAN)
The MAN (F4) key pauses the running program at the end of the current instruction. However, a motion is immediately stopped to zero-velocity by the normal ramp-down process, i.e. the true position is preserved. The START button is blinking and the 'MAN' is displayed over F4 key. To resume execution, press START again.

STOP
A first depression of STOP while a program is running immediately stops the execution. The current motions are ramped to 0-velocity, the outputs and the DAC are reset. The true positions of the axis are preserved. Execution may be resumed by depressing START. The first depression of STOP has the same action as the MAN key. A second depression aborts the current program. The pilot light of STOP is on and the program which will come to execution when depressing START is the "POWER ON PROGRAMME".

After switching the controller on, the mode is MOD1 and the POWER ON PROGRAMME is automatically executed. If a power on program is not wanted, enter 100 as power on vector.

Note:
Most utilities are available while a program is running. However, the editor should be used with care: a line insertion or deletion will move the portion of the user's memory above the current line. Catastrophic failures may result.

7.3 Fault Processing

Two fault situations are processed by UNIPROG:

1. The fault generated within the motor drivers,

2. The over-travel as detected by software limits, (only in contouring mode, in positioning and vector modes, the travel is a priori limited)

When a fault situation arises, UNIPROG immediately stops all motions, all outputs and the DAC. The screen shows one of the messages:

```
AXIS a fault
press STOP
```

```
STROKE a TOO LARGE
press STOP
```

where "a" stands for the axis, X, Y.

Depressing STOP resets, the screen shows:

```
AXIS a FAULT
JOGGING +/-  -> ESC
```

The faulty axis can be moved slowly with the JOG keys. This is useful when the driver is fitted with hard-wired limit switches. The ESC key then returns the controller to the normal status. If the fault remains, some thing must be wrong with the driver.

In any case the controller executes the POWER ON PROGRAMME after a recovery from a fault situation.

CAUTION:
The red STOP button is by no means an emergency stop as required by the regulations.

8 Vector Generation and Contouring

8.1 Features and Space Definition

The E-300 Series Controllers are able to generate vectors in X and Y dimensions. Contouring pathes are obtained through the generation of small straight segments. The basic PINX-E language allows the programming of pathes of any shape in Cartesian, polar or cylindrical coordinate systems.

UNIPROG has a set of instruction to generate straight and circular pathes in **Cartesian coordinates**. The programming task is then a lot easier.

The programmer has to make the difference in generating a single straight vector or generating a continuous path composed of several straight and circular segments. A single vector is produced whenever a positioning instruction is written with the execution mode 3 (3 = /). The geometry of a continuous path must be defined with pseudo-instructions outside the executable program.

While generating a vector or a continuous path, the velocity is controlled along the **path ordinate**. The path ordinate plays the role of virtual axis whose parameters are derived from the involved axes.

A path with abrupt changes of direction induces discontinuities in the velocity of the axes. The step motors will not necessarily be able to follow the path at any speed.

8.2 Vector Generation

40 ORGP <*axis*> <*absolute position*>

The instruction ORGP (**ORiG**in **P**ath) sets the reference of the contour and must follow the instruction DPATH. If the reference of the contour is located at the starting point, the instruction ORGP is unnecessary.

30 DPATH <*space*> <*tool left-right*>

The instruction DPATH (Define PATH) determines the plane in which the contour will be created. The parameter left-right (LR) indicates toward which side the correction of the tool trajectory should go.

- **This instruction must imperatively appear in the first line of the contour file.**
- The designated space is active until a new DPATH instruction is encountered.
- A vector motion requires the mode 3 in the POSA/POSR instructions.

Three important points must be emphasized:

a) That latent motions (set-up with POSR with mode 0) will be executed, even on axes outside the defined space.

b) The home (reference) position must be done on **all axes** prior to any vector generation in mode 3.

c) The travel will be automatically limited to the value "STROKE", see section 4.4.1.

8.3 Programming the Geometry of a Continuous Path

A continuous path is a concatenation of straight, and circular segments executed as single move at a constant path velocity. The description of the geometry of the path is written in a file outside the executable program. A geometry file can hold several pathes. The limited

computing power of the E-300 controller is overcome by a pre-interpretation of the geometry. The programmer has to instruct the path interpretation as described in section 8.4.

A path is defined in its own coordinate system, without any relation to the coordinate systems of chapter 3. The starting point of a path is the origin of the coordinates. While executing the path motion, the path starts at the actual axis position. Thus, geometry file can be "re-used" at several positions.

The pseudo-instruction DPATH is mandatory at the beginning of a geometric path definition. (DPATH is also an executable instruction, refer to section 8.2). When several pathes are described in a single file, they are separated by the DPATH pseudo-instruction. The END directive must be used only at the very end of the geometry file.

8.3.1 Definition of a Straight Segment

32 LINA <*axis*> <*coordinate*> <*mode-e*>

33 LINR <*axis*> <*component*> <*mode-e*>

The pseudo-instructions LINA and LINR behave similarly to the instructions POSA and POSR, "mode-e" parameter has the same meaning as in these last instructions (see chapter 5.1.2). The concepts of the absolute coordinate and relative displacement refer to the space of the contour, in Figure 8-1. and not to the reference in chapter 3. A pseudo-instruction is required for each component of the vector which can be given in an unspecified order. The last pseudo-instruction must be in mode 2.

The coordinates or the components are naturally given in the unit defined by SCALEK. Mode 0 applies to all the components except the last one, which must be in mode 2.

If one of the vector's coordinates or one of its components has not been specified, the last-mentioned axis is taken into account. This characteristic allows easier and more concise programming of the tabulated functions as shown in the following example. This simplification no longer functions after programming an arc of a circle nor after the instruction POINT.

The instructions LINA and LINR must always follow the instructions POINT see (POINT).

```
DPATH          XY      L
LINA    X      3 0
LINA    Y      2 2     ; seg. 1
LINA    X      5 2     ; seg. 2
LINA    X      6 0
LINA    Y      3 2     ; seg. 3
LINA    Y      5 2     ; seg. 4
.....
.....
```

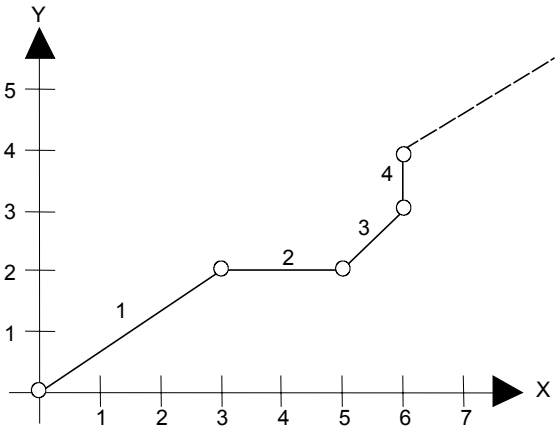


Figure 8-1 : Contour with zero diameter

If a diameter of the tool correction is required, it is necessary to use the instruction POINT with an arc of zero radius (RAD = 0), instead of the instructions LINA or LINR.

```
DPATH       XY     L
POINT X     3 0
POINT Y     2 2    ; seg. 1
POINT X     5 2    ; seg. 2
POINT X     6 0
POINT Y     3 2    ; seg. 3
POINT Y     5 2    ; seg. 4
.....
.....
```

42 POINT <*axis*> <*absolute position of the angle*> <*mode-e*>

The instruction POINT makes it possible to mark the coordinates of the angle in which will be stored the radius of the rounding determined by the instruction RAD.

The coordinates of the summits are given in the reference of the contour, i.e. in relation to an unspecified point of the surface of the contour. In the case of a figure representing central symmetry, the coordinates will be given, naturally, with respect to the center.

In a contour defined by several POINT instructions, it is not necessary to recall the last coordinate if it is identical.

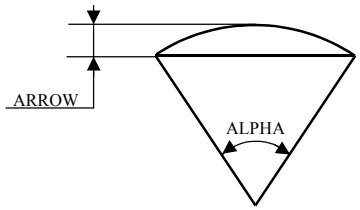
To complete a contour defined by the POINT instructions, the file must be terminated by the instruction LINA or LINR. The 2 coordinates of the surface must be recalled to terminate the contour.

8.3.2 Definition of a Circular Segment

A polygonal approximation is used to generate a circular segment. The angular definition of this approximation is given by the pseudo-instruction CDEF (Circular Definition). CDEF is a modal directive, i.e. the angular definition is valid until changed by a new CDEF.

35 CDEF <*maximum arrow (tolerance)*>

The instruction CDEF determines the angle of segmentation to obtain the maximum acceptable deviation of a segment for each subsequent circle.



34 RAD <*mode-r*> <*radius*>

"mode-r" parameter of the instruction RAD (see Figure 8-2), associated with instructions CIRR and CIRA, determines on which plane the circle must be created and with which radius. However this instruction becomes unnecessary if the circle follows another circle or a straight line. The program UNIPROG+ can automatically generate the radius and the mode while respecting the contour without angular points.

The instruction RAD associated with the instructions POINT makes it possible to generate the radius of the rounding. In this case the mode is unnecessary.

If the radius is identical, it is not necessary to recall it for subsequent circles.

A circle's arc is determined by the following elements:
- components of the cord under-drawn by the arc or the coordinates at the end of the arc,
- the radius of the circle,
- the mode specifying one of the 4 possible solutions, see Figure 8-2.

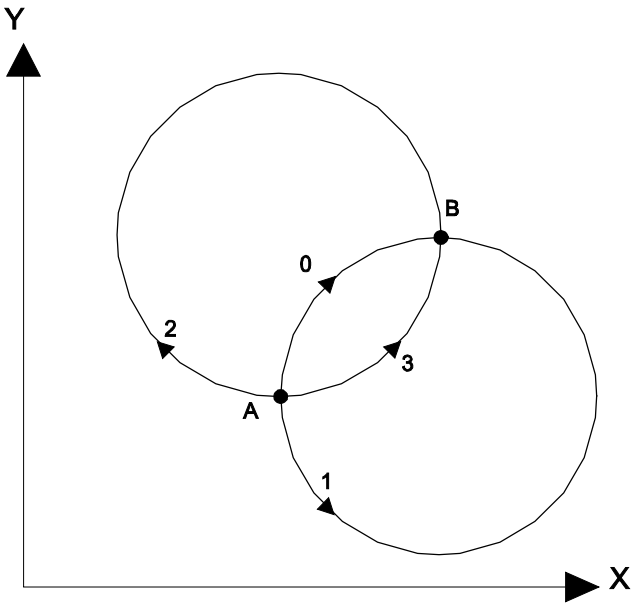


Figure 8-2 : Rotation Modes

The radius and the mode are compulsory on the first arc of the circle. For the following calculations, they become optional in the case of a contour without angular points. In fact, UNIPROG+ is able to determine the radius automatically.

36 CIRA <*axis*> <*coordinate*> <*mode-e*>

37 CIRR <*axis*> <*component*> <*mode-e*>

The pseudo-instructions CIRA and CIRR behave like LINA and LINR. If several arcs of the circle in the same contour have the same radius and the same mode, the pseudo-instruction RAD can be written only once. CDEF and RAD must precede CIRA/CIRR.

As an example, let us give two ways of coding of the contour of Figure 8-3.

```
DPATH     XY    L                 DPATH     XY    L
LINA      X     3 0               LINR      X     3 0
LINA      Y     2 2               LINR      Y     2 2
RAD       3     2.5               RAD       3     2.5       ;optional
CIRA      X     4 0               CIRR      X     1 0
CIRA      Y     4 2               CIRR      Y     2 2
RAD       0     5                 RAD       0     5         ;optional
CIRA      X     6 0               CIRR      X     2 0
CIRA      Y     7 2               CIRR      Y     3 2
END                               END
```

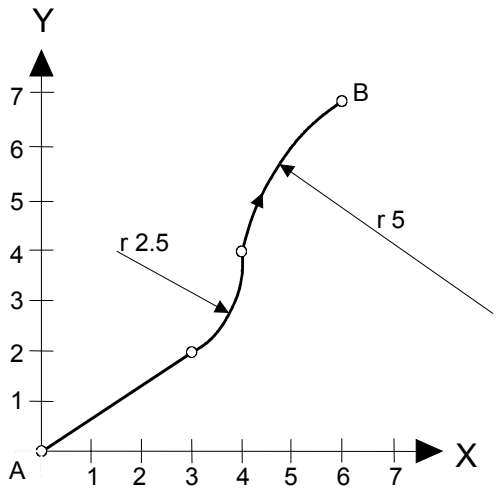


Figure 8-3 : Contour Example

8.4 Interpretation of the Geometric Files

As already mentioned, the geometric files must be computed prior to execution. The instruction PCOMP generates for each path a buffer of numerical data to be used in real time by the motion generators. This operation is rather time consuming, thus it must be done only once at power-up or during idle time.

Two situations may arise:

- There is enough space available in the memory to store all the buffers used by the program. The PCOMP instructions will all be located in the POWER ON PROGRAMME. A path computation takes place after switching on or after a full stop by two depression of the STOP button.
- The available space in memory is too small; it is then necessary to organize the computation in such a way that the results will be available early enough and that the computation time will not be noticed. PCOMP writes its results on a rotary buffer; thus, care must be exercised not to overwrite valuable data. Please, consult E.I.P.SA to solve specific problems related to PCOMP.

31 PCOMP <*file*>

The argument "file" is the file number where the geometric definition is stored. If there are several pathes separated by DPATH directives, the interpretation goes till the end of the file

47 TOOLP *<tool number> <number of the contour file>*

The instruction TOOLP sets a new reference and loads the contour file associated with this tool for all the instructions of positioning and subsequent contouring. The components of the translation vector of the origin are stored in file "0".

- The " TOOL " table can allow consultation or modification of these origins as well as the diameter of the associated tool.
- These origins can also be memorized and validated directly from the jogging.
- The parameter " tool number " selects the group of the 4 components associated with a tool number.
- When the program executes the instruction TOOLP, the red LED “STOP” blinks indicating that the contour file is being calculated. The contour file is not recalculated as long as it is not modified.
- The contour uses the tool number to determine the radius of the correction of the tool trajectory.
- The radius is stored in the table " TOOL ".

8.5 Execution of a Path

48 PATH *<speed>*

The file containing path dates was loaded by the last TOOLP instruction.

If the instruction PATH is **directly** followed by the WAITP instruction then the contour starts without waiting for the end of the latter instruction. To avoid execution of disordered sequences of movement, the rest of the program must imperatively contain the parameter indicating the end of the contour (instruction 66 ENDP), before any other new movement can be begun.

66 ENDP

Parameter for the end of a contour (END Path) to be used imperatively when the instruction PATH is directly followed by the instruction WAITP.

82 WAITP *<axis> <speed> <position> <mode-w>*

Waits if the absolute position of the concerned axis is smaller or larger according to the mode.

Mode-w = 0	Waits as long as the absolute position is smaller than the value of the programmed position.
Mode-w = 1	Waits as long as the absolute position is larger than the value of the programmed position.

- The programmed position takes into account the origin of the tool but not the origin of the contour file.
- This instruction is placed directly after the instruction PATH, without this condition the contour is executed until the end.
- The reaction time of this instruction depends on the number of simultaneous functions active in the command.
- If the programmed position is not reached the program must be stopped with the STOP key.

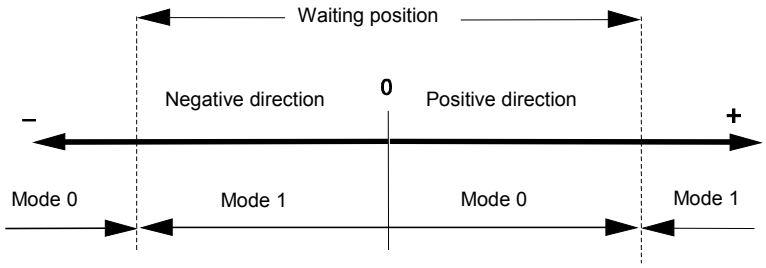
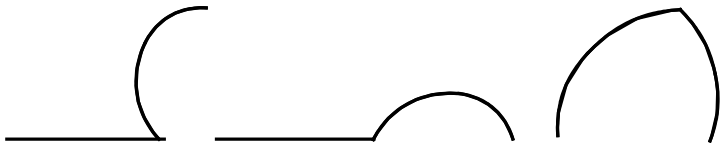


Figure 8-4 : Waiting position according to the displacement direction

8.6 Case not accepting the correction of the Tool

UNIPROG+ does not solve all cases of contournage, in particular when discontinuities or summits appear in the contour.
To solve these cases, it is necessary to create a linear tangent segment with the arc (minimum length of 0.02 mm).

For example a straight line cutting an arc or 2 non-tangent arcs.



8.7 Display of the Contour Errors

During the pass of the calculation of the contour numbered ERRORS can appear in the case of erroneous data. These errors stop the calculation.

Error 0:	The radius is negative.
Error 1:	The contour is impossible. The coordinates at end of the arc (CIRR CIRA) are beyond the radius. The radius is negative.
Error 2:	Division by 0 during the use of the instruction POINT. The radius is negative.
Error 3:	Division by 0 during the automatic generation of a tangent arc to another arc.
Error 4:	Inaccuracy in the calculation of the angle.
Error 5:	Inaccuracy in the calculation of the center.
Error 6:	The determination of the departure point of the contour is impossible. The first segment (LIN) has to measure more than 0.01 mm.
Error 7:	Instructions LINR or LINA should not precede the instruction POINT.
Error 8:	2 linear segments follow on the same axis (LINR, LINA). The contour can have a discontinuity. This case is only accepted when the radius of the tool is zero. The use of the instruction POINT with a zero radius can compensate for this disadvantage.
Error 9:	Division by 0, the contour is impossible, the coordinates of the circle are beyond the radius.
Error 10:	The instruction RAD is missing. In concrete terms, determining the radius is impossible when the contour starts with a circle.
Error 11:	The coordinates at the end of the arc conflict with those at the point of departure.
Error 12:	The generation of the rounding is impossible, the 3 co-ordinates forming the angle of the polygon are aligned, consequently the radius is infinite.

Error 13: There is no contour file available.

Error 14: The instruction DPATH is missing in the first line of the contour file.

Error 15: At the time of execution, the deposit of the contour (BUFFER) is outside of the memory zone. A false manipulation has modified the first line of the contour file.

Error 17: The instructions POINT do not follow the instructions LINA or LINR.

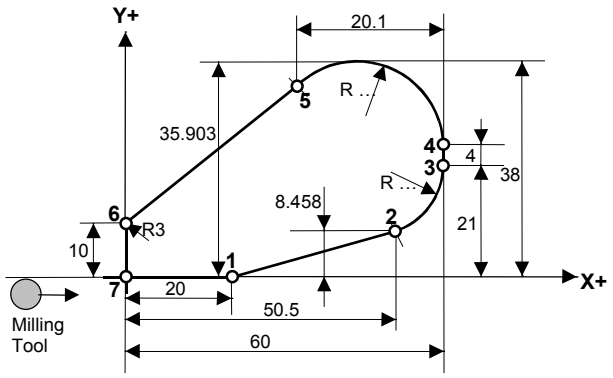
RAYON < 1: The contour presents at least a radius smaller than 1 mm. Consequently the execution speed of the contour will have to be adapted to run the radius.

NULL SEGMENTATION: The arc is so short that the segments to the right cannot be reached. It therefore generates a straight line on the coordinates of the end of the arc.

8.8 Examples

" TOOL " tables

ORIGIN X	TOOL 0	?
ORIGIN Y	TOOL 0	?
DIAMETER	TOOL 0	?
ORIGIN X	TOOL 1	13
ORIGIN Y	TOOL 1	54
DIAMETER	TOOL 1	8



File 1

0	01	47	TOOLP	1	23		;Charges the origin of tool 1 of contour file 23
							;Origin X 13 and Y 54, tool diameter 8 mm taken from the table TOOL
1	01	46	CORR	2			;Places the tool on the corrected contour, speed 2
2	01	48	PATH	1			;Executes contour 23 loaded by TOOLP, speed 1
3	01 *	10	POSA	X	0 50.0000	0	
4	01	10	POSA	Y	0 50.0000	2	;frees the tool, speed 0

File 23 sub program contour

0	23	30	DPATH	XY	L		;Defines the XY space and the correction to the left " L "
1	23	40	ORGP	X	-1.0000		;Begins the contour in recess from X of 1 mm
2	23	35	CDEF		0.01		acceptable error on the arc 0.01 mm
3	23	42	POINT	X	20.0000	0	
4	23	34	RAD	?	0.0000		;Radius of rounding is zero, mode not necessary
5	23	42	POINT	Y	0.0000	2	;1st point, angular
6	23	32	LINA	X	50.5000	0	
7	23	32	LINA	Y	8.4580	2	;2nd point, tangent with the arc
8	23	36	CIRA	X	60.0000	0	
9	23	36	CIRA	Y	21.0000	2	;3rd point, end of the arc, automatic generation of the arc radius
10	23	33	LINR	X	0.0000	0	
11	23	32	LINR	Y	4.0000	2	;4th point, tangent with the arc
12	23	37	CIRR	X	21.100	0	
13	23	36	CIRA	Y	35.9030	2	;5th point, end of the arc, automatic generation of the arc radius
14	23	42	POINT	X	0.0000	0	
15	23	34	RAD	?	3.0000		;Radius of the rounding 3 mm, mode not necessary
16	23	42	POINT	Y	10.0000	2	;6th point angle of the rounding
17	23	32	LINA	X	0.0000	0	
18	23	32	LINA	Y	-2.0000	2	;7th point, end of the contour in recess from Y of 2 mm

8.9 Summary of Contouring Instructions and Pseudo-Instructions

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description
35	**CDEF**	Max. arrow				Define segmentation value
36	**CIRA**	Axis	Coordinate	Mode-e		Define absolute circular mvt
37	**CIRR**	Axis	Component	Mode-e		Define relative circular mvt
30	**DPATH**	Space	Tool L-R			Define working Plan
66	**ENDP**					End of path
32	**LINA**	Axis	Coordinate	Mode-e		Define absolute straight segm.
33	**LINR**	Axis	Component	Mode-e		Define relative straight segm.
40	**ORGP**	Axis	Abs. Pos.			Set Path origin
48	**PATH**	Speed				Path execution
31	**PCOMP**	File number				Geometric file interpretation
42	**POINT**	Axis	Angle pos.	Mode-e		Segm. bounded with rounding
34	**RAD**	Mode-r	Radius			Define radius and rotation dir.
47	**TOOLP**	Tool Nr	File Nr			Geometric file interpretation
82	**WAITP**	Axis	Speed	position	Mode-w	Waiting reaching a position

9 UNIPROG Recapitulation

9.1 Instructions:

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description	Page
91	**ADD**	address				Add to Accu Direct	34
86	**ANGLE**	Speed	Value	Mode-e		Displ. angle (polar coordinate)	27
22	**BRIN0**	input	address			Branch if Input is False	29
23	**BRIN1**	input	address			Branch if Input is True	29
24	**BRM**	address				Branch if Accu is <0	34
27	**BRNZ**	address				Branch if Accu is non zero	34
25	**BRP**	address				Branch if Accu is >=0	34
26	**BRZ**	address				Branch if Accu is zero	34
61	**CALL**	address				Sub-Routine Call	33
35	**CDEF**	Max. arrow				Define segmentation value	42
36	**CIRA**	Axis	Coordinate	Mode-e		Define absolute circular mvt	43
37	**CIRR**	Axis	Component	Mode-e		Define relative circular mvt	43
18	**CLOS**	axis	speed			Closure Check	25
46	**CORR**	Speed				Rectilinear displacement	27
95	**CPL**	Output Nr				Output state complement	31
59	**DECD**	address				Decrement Address Direct	32
79	**DISPD**	Position	Address			Display a value	31
94	**DIVD**	address				Divide Accu Direct	34
30	**DPATH**	Space	Tool L-R			Define working Plan	40
62	**END**					End of Prog. and Routines	33
66	**ENDP**					End of path	45
65	**ENDRP**					End of Repeat Loop	33
98	**FDATA**	floating				Define a Floating Number	34
50	**FLOAD**	number				Load Accum Immed.,Floating	32
99	**IDATA**	integer				Define an Integer Number	34
51	**ILOAD**	integer				Load Accum Immed.,Integer	32
58	**INCD**	address				Increment Address Direct	32
60	**JMP**	address				Unconditional Jump	33
32	**LINA**	Axis	Coordinate	Mode-e		Define absolute straight segm.	41
33	**LINR**	Axis	Component	Mode-e		Define relative straight segm.	41
52	**LOADD**	address				Load Accum Direct	32
53	**LOADI**	pointer				Load Accum Indirect	32
85	**MOTOR**	Motor Nr	Rot. Speed			Motor rotation speed	32
93	**MULD**	address				Multiply Accu Direct	34
90	**NOP**					No Operation	34
28	**OFF**	output				Set Output to 0	29
29	**ON**	output				Set Output to 1	30
88	**ORGA**	Angul. shift				Angular shift of reference axis	28
40	**ORGP**	Axis	Abs. Pos.			Set Path origin	40
48	**PATH**	Speed				Path execution	45
31	**PCOMP**	File number				Geometric file interpretation	44
84	**PECK**	Axis	Slow speed	Drill pos.	Mode-d	Peck cycle (drilling)	26
42	**POINT**	Axis	Angle pos.	Mode-e		Segm. bounded with rounding	42

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description	Page
10	**POSA**	axis	speed	coordinate	Mode-e	Absolute Indexing,Immed.	24
11	**POSAD**	axis	speed	address	Mode-e	Absolute Indexing,Direct	24
12	**POSAI**	axis	speed	pointer	Mode-e	Absolute Index. Indirect	24
14	**POSR**	axis	speed	disp'ent	Mode-e	Relative Indexing,Immed.	25
15	**POSRD**	axis	speed	address	Mode-e	Relative Indexing,Direct	25
16	**POSRI**	axis	speed	pointer	Mode-e	RelativeIndex. Indirect	25
34	**RAD**	Mode-r	Radius			Define radius and rotation dir.	43
87	**RADIUS**	Speed	Value	Mode-e		Displ'mt. radius (polar coordinate)	28
90	**RBW**					Restore basic window	31
17	**REF**	axis				Reference Point	25
63	**REP**	n times				Repeat, Immediate n	33
64	**REPD**	address				Repeat, Direct n	33
54	**SAVE**	address				Save to EEPROM	32
83	**SET**	Param. Nr	Param. Val.			Setting of variables	26
67	**SIM1**	address				1st Simultan. Prog. Call	33
68	**SIM2**	address				2nd Simultan. Prog. Call	33
57	**SPVEL**	Rot/min				Spindle speed	32
55	**STORD**	address				Store Accum Direct	32
56	**STORI**	pointer				Store Accum Indirect	32
92	**SUBD**	address				Subtract from Accu Direct	34
13	**TEACH**	axis	speed	address		Teach-In,Direct Argument	25
19	**TOOL**	Tool numb.				Set Tool parameters	25
47	**TOOLP**	Tool Nr	File Nr			Geometric file interpretation	45
81	**TPING**	Axis	Pitch	Tap. Pos.		Tapping instruction	27
70	**WAIT**	time				Dead Timer, Immed. Time	34
20	**WAIT0**	input				Wait if Input is False	29
21	**WAIT1**	input				Wait if Input is True	29
71	**WAITD**	address				Dead Timer, Direct Time	34
82	**WAITP**	Axis	Speed	position	Mode-w	Waiting reaching a position	45
89	**ZTOOL**	Axis	Input nb	direction		Ref. for auto. tool adjustment	28
72	**2HON**					Beginning of 2 hand start section	
73	**2HOFF**					End of 2 hand start section	

Table 9-1 : UNIPROG+ Instructions

Axes: X = 0 Y = 1

9.2 Inputs and outputs:

Input	Item	Output	Item
0	IN(0)	0	OUT(0)
1	IN(1)	1	OUT(1)
2	IN(2)	2	OUT(2)
3	IN(3)	3	OUT(3)
4	IN(4)	4	OUT(4)
5	IN(5)	5	OUT(5)
6	IN(6)	6	OUT(6)
7	IN(7)	7	OUT(7)
8	SIM(0)	8	SIM(0)
9	SIM(1)	9	SIM(1)
10	SIM(2)	10	SIM(2)
11	FLAG(0)	11	FLAG(0)
12	FLAG(1)	12	FLAG(1)
13	FLAG(2)	13	FLAG(2)
14	FLAG(3)	14	FLAG(3)
15	FLAG(4)	15	FLAG(4)
16..49	IN(16)..IN(49)	16..63	OUT(16)..OUT(63)
50..59	KEY("0".."9")		
60	INA0		
61	INB0		
62	INA1		
63	INB1		

Table 9-2 : UNIPROG+ Inputs and Outputs

10 E300 Wiring

10.1 Compact Controller Type E300-CMP

10.1.1 Compatibility with E-600

The I/O EXT connectors and E600-3 connectors are same between E300-CMP and E600-Base. The I/O connector is similar but not fully compatible with E600. Therefore, plugging of E600 cable into E300 connector is not destructive. The position of some signals is the same as E600.

Following table show the differences between I/O connectors:

E300 and E600 I/O 19 pin comparison		
Pin	E300	E600
A	0V	0V
B	OUT4	OUT4
C	OUT5	OUT5
D	OUT6	OUT6
E	OUT7	OUT7
F	IN0	Analog GND
G	IN4	DAC out
H	OUT0	OUT0
J	IN1	+5VDC (output)
K	IN5	ADC input
L	IN2	IN2
M	IN6	IN6
N	IN3	IN3
P	IN7	IN7
R	0V	AGND
S	OUT1	OUT1
T	OUT2	OUT2
U	OUT3	OUT3
V	+24VDC	+24VDC

Tableau 10-1 : E300 et E600 I/O comparison

The ANALOG I/O doesn't exists in E600. The RS-232 pinning is not the same as E600.

10.1.2 I/O Connector

The I/O connector regroups 24 VDC inputs and outputs, and 24V supply for them.

Pin	Signal
A	0 V, output return
B	OUT(4), 24 V, 1 A
C	OUT(5), 24 V, 1 A
D	OUT(6), 24 V, 1 A
E	OUT(7), 24 V, 1 A
F	IN(0) 24V input
G	IN(4) 24V input
H	OUT(0), 24 V, 1 A
J	IN(1) 24V input
K	IN(5) 24V input
L	IN(2) 24V input
M	IN(6) 24V input
N	IN(3) 24V input
P	IN(7) 24V input
R	0 V, output return
S	OUT(1), 24 V, 1 A
T	OUT(2), 24 V, 1 A
U	OUT(3), 24 V, 1 A
V	Unregulated +24 VDC supply

Tableau 10-2 : E300 I/O Connector, 19 pin Burndy

Each output can deliver 1A but the sum of the 8 output currents must not exceed 4 ampere.

10.1.3 I/O EXT Connector

This connector regroups the signals for external I/O modules like E-500-I1, I2, I3 and E-500-ODC1.

10.1.4 RS 232 Connector

RS-232 connector is designed to connect the E300 to a PC through a 1 to 1 cable, for use the NewWincom or APEX software.

10.1.5 E-600-3 Module, 2 Phase Step-by-Step Motor Translator from EIP

"Slow/fast decay" system translator, with 1600 microsteps.

Pin	Signal
A	B phase winding
B	B phase winding
C	A phase winding
D	A phase winding
E	INA 24V input
F	INB 24V input
G	+24VDC supply
H	0V

Tableau 10-3 : E600-3 Connector, 8 pin Burndy

10.1.5.1 Current Setting

The rotative selector is designed to choose the peak to peak current corresponding to the motor. The value of the current is obtained when the BOOST signal is active. Otherwise the current is

reduced to 60% of the selected value.

Position	Current	Position	Current
0	2.0 A	5	5.3 A
1	2.7 A	6	6.0 A
2	3.3 A	7	6.7 A
3	4.0 A	8	7.3 A
4	4.6 A	9	8.0 A

Tableau 10-4 : E600-3, Current Setting

10.1.6 ANALOG I/O Connector

This connector regroups analog inputs and outputs. This is a D-sub 9 pole male type:

Pin	Description	Remarque
1	+5Vref	OUT
2	ADC1	IN
3	ADC2	IN
4	DAC0	OUT
5	DAC1	OUT
6	AGND	-
7	AGND	-
8	AGND	-
9	AGND	-

Tableau 10-5 : E300 Analog I/O connector