E I P **E-600**

English

UNIPROG+ V7-3x

General Purpose Programming Language
For the E-600 Motion Controller

Version: **7 March 2002**

E I P

UNE GAMME COMPLETE DE CONTROLEURS D'AXES
EINE VOLLSTANDIGE PALETTE VON ACHSENSTEUERUNGEN
A COMPLETE RANGE OF MOTION CONTROLLER

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-600 Motion Controllers are aimed at the market segment: special machine tools, handling equipment's and assembly automation. Intricate motion control problems and elaborate sequencing can be solved easily. The E-600 controllers are available for 1 to 4 axes, with step-motor translators for 2 and/or 5 phase motors. The controller also has provision for two-closed loop drives.

The external connections are to a large extent compatible with the E-500 series, however, the format of the storage card is not directly compatible. **A conversion of E-500 cards to E-600 is ready made with the UNICOM Program.**

"**E-600 base**" is the designation for the controller with the exception of the power drivers. It includes the power supply, the control logic, the keyboard and the local inputs and outputs. The housing has space for various drive modules. We also manufacture a version of the E-600 controller, the "**E-600-ND**" (No Drive) intended to be incorporated into a customer enclosure.

E.I.P. 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 user's programs and the machine configuration may be saved in a removable memory card (BEE-CARD). This EEPROM card is a very practical and reliable storage means.

1.1 About UNIPROG+

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.

- UNIPROG+ V 7.30 has been designed to replace two older programs, UNIPROG and POLYTOOL, while maintaining a similar look and feel. It offers the advantage of accepting a trajectory corrector of the tool radius and the contours (tool contours) (instructions LIN, CIR, NOT).
- The primary restriction being that the contour must be presented without angular points all the segments and arcs being tangents.
- This inconvenience is compensated by the fact that the radius and mode can be automatically generated when several arcs are chained together; therefore rendering the RAD instruction becomes unnecessary.
- To simplify the programming aspect, UNIPROG+ also makes it possible to program angular points with their rounded edges (previously POLYTOOL).
- During the calculation phase of the contour a variety of messages can appear indicating certain contouring errors. In addition, in mode 2, the display indicates the radius, which define the contour.

To install UNIPROG+ on an E-600 controller already equipped with the software UNIPROG or POLYTOOL, please read the chapter COMPATIBILITY WITH PREVIOUS SOFTWARE VERSIONS if older parts programs must be preserved.

1.2 Compatibility with UNIPROG

1) If, in older files, the instruction 19 ORG is not used, the parameter " LAST TOOL NB " must be put to zero. On the other hand, if instruction 19 ORG is used, it is necessary to release

file " 0 " which will then be used to deposit the origins of the tool. Consequently it is necessary to copy it to another file and to modify the addresses which refer to file " 0 ".

2) To obtain the value of the DAC in % the parameter " MAX RPM 10 volts " is set to 100. Furthermore in the program, the association of the instructions FLOAD and LDAC is replaced by the instruction 57 SPVEL that is then followed by the rotation speed.

3) The angular definition given by the instruction CDEF will have to be replaced by the maximum acceptable error.

4) To obtain START on one input only, the parameter " 2 HAND START " must be set to 8.

5) To avoid calling up programs during jogging, the parameter " MANUAL PROGRAMME " must be set to 100 (see CTRL and JOG).

1.3 Compatibility with POLYTOOL

1) File " 0 " which is used to deposit the origins of the tool passes from 4 to 5 values per tool. Consequently, the origins of the tool will be shifted.

2) See paragraphs 2, 3, 4, 5 above for compatibility with UNIPROG.

3) The first two POINT instructions will be replaced by instructions 40 ORGP.

4) Code 32 of the instruction POINT becomes code 42 and mode 2 will have to be added to the second coordinate.

5) The instruction DTOOL disappears and the parameter left/right is added to the DPATH instruction. It is necessary to specify the diameter stored in the TOOL table. The diameter is associated with the tool number chosen by the instruction TOOLP.

6) The parameter TOOL L/R INVERT must reflect the directions of the axes.

7) The contours are always opened and terminated by the instructions LINA or LINR.

1.4 About this Manual

The aim of this manual is to allow the inexperienced user to master UNIPROG after a thorough reading. Some knowledge of step motor techniques and a practical experience with servo drives are 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.
- The section 8 deals with the problem of vector and path generation (Linear and circular interpolation).

2 Memory Organization

2.1 Physical Memories of the E-600 Controller

The memory space is made of 4 different chips:

- A 32 kbyte EPROM, not field alterable, 16 kbytes are available within the EPROM to store the "lower part " of the system program,
- A 32 kbyte FEPROM (FLASH-EPROM), field alterable, intended to store the "higher part" of the system program,
- A 32 kbyte RAM with battery back-up,
- A 8 kbyte EEPROM on a removable card, the "Data Card".

Utilities are provided to save portions of the RAM into the Data Card or to dump the card into the RAM.

The FEPROM can be loaded from a special card plugged into the slot (SRAM-Card or EPROM with an adapter) using a hidden utility. This process is described in the Hardware Manual. Thus, a controller can be reconverted in the field to run other system programs.

The Data-Card (EEPROM) is used to save user's programs written in UNIPROG language and the configuration of the whole equipment.

The live memory (C-MOS RAM) is the current working area. If a Data-Card is plugged at the power-up, the whole contents of the card is dumped into the RAM. An UNIPROG program is always executed from the RAM. The editor and the configuration utilities write to the RAM, never to the Data-Card. Saving edited programs or new configuration data has to be done by the operator.

2.2 The User's Storage Area

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

Within the user's area, a fixed portion is reserved to the configuration parameters, see section 4.4. The remainder (7500 bytes) is available for programs or numerical data.

The UNIPROG editor stores **"lines"** (6 bytes). An instruction or a numerical data are always stored as one line. 1250 lines are available; 100 files may be opened within the line set. The files are numbered 00 to 99. A file is opened by the editor or by the copy of an existing file.

Of course, the user's storage area can be extended simply by an other Data-Card.

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 or the servo drive with an incremental encoder requires a home position before starting any useful work. Two different situations are common:

a) The machine has its own coordinate system, such as a jig-boring machine for example.

b) 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 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

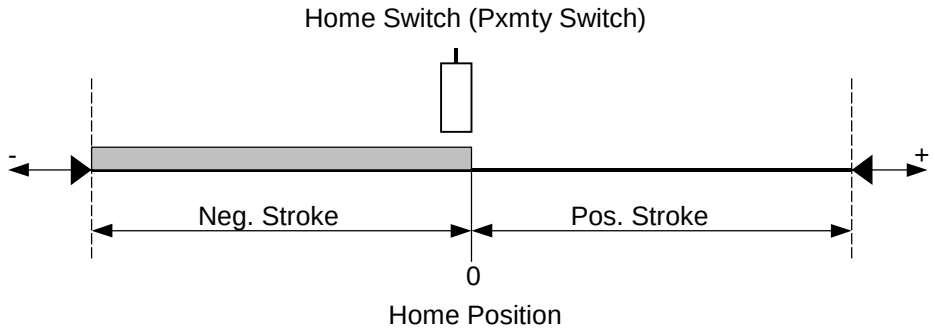


Figure 3-1 : Travel and Home Position

The process of finding the home position has three phases:

Phase 1 : (This phase takes place only if the home switch is active while entering the process.) The slide walks out of the home switch.

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 stop 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-600 motion controller has 8 inputs, which can be used as Home -or Reference-switches: REF INPUT 0..7. The assignment of a particular input to a given axis is also a configuration parameter.
- If an axis does not use a home switch, it must be assigned conventionally to REF INPUT 8.

The origin of the initial coordinate system will be fixed by the execution of home function without any motion.

- With servo drives, the index channel of the encoder is normally used for an accurate home position. A rough "home" switch on the slide is still required. In this case, the REF INPUT assignment is from 9 to 17:

- REF INPUT 9 is used with the servo modules E-6004, E-600-7 and E-600-12 when the home position is at one travel limit. The controller makes use of the fault signal generated by the limit switches (and the index pulse of the encoder) to determine the home position.

- REF INPUT 10 to 17 are used with the above mentioned servo modules with a switch on the slide wired to one of the inputs LS(0) to LS(7).

- With the adapter for YASKAWA servo drives, the home switch has a fixed assignment: REF INPUT 0 for axis X, 1 for Y, 2 for Z and 3 for U. To use the limit switch as an home switch, REF INPUT 18 must be inputted. The corresponding LS-input (LS(0) for axis X, LS(1) for Y...) is no more available for other purpose.

REF INPUT	Module Type	Physical Input	Home Switch Arrangement
Equal to Axis Number	E-600-8	LS(0) to LS(3)	Home Switch and Index
0 to 7	All	LS(0) to LS(3)	Index not Used
8	All	NA	(No Motion)
9	E-600-4/7	NA	Limit Switch and Index
10 to 17	E-600-4/7	LS(0) to LS(7)	Home Switch and Index
18	E-600-8		Limit Switch and Index

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

- The sign of REF SPEED in the configuration governs the direction of the motions involved in the determination of the home position.

- 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 REF INPUT 8 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.

UNIPROG has provision for coordinate translation, a coordinate rotation is not available.

For an ultimate flexibility, two levels of coordinate translation are provided:

- The OFFSET vector transforms the **Home Coordinate System** into the **Base Coordinate System**,
- The ORIGINE vector translates the Base into the **Current Coordinate System**, see Figure 3-2.

The Home System is not directly addressable, unless OFFSET = 0, i.e. the Home and the Base Systems are superimposed.

The OFFSET vector is part of the configuration. It is useful whenever the natural zero of the coordinates differs from the home position, examples: Jig-boring machines, stamp presses...

The JOGGING utility uses the Base Coordinate System to effect absolute moves.

The Current Coordinate System is set-up by the instruction ORG. The programmer may change the current system at will. If no ORG 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.

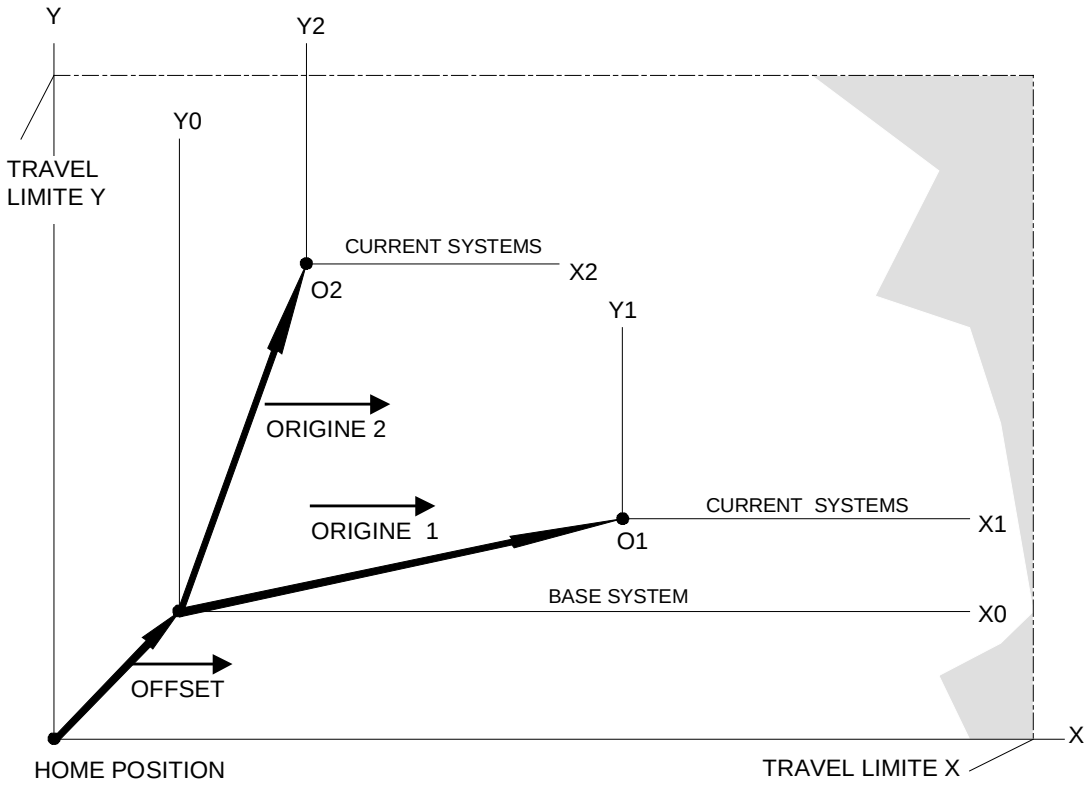


Figure 3-2 : The UNIPROG coordinate Systems

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.

For a servo drive, SCALEK is 4 times the number of pulses generated by the encoder during a travel of one length unit. (4 times, because all transitions of the quadrature signals are getting counted)

The micro-stepping 2-phase translators E-600-3 needs 8 pulses for one full step. With the most usual step motors (1.8 degree/step), 1600 pulses produce exactly one revolution.

Berger 5-phase motors and drivers require 500 or 1000 pulses per revolution, depending on the settings on the board.

Examples:

a) 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

b) Rotary Table:

5-phase motor 1000 pulses per motor revolution,
motor directly drives worm gear 1:40,
unit: 1 degree.

40'000 pulses for one revolution of the table,
40'000/360 pulses per degree, SKALEK = 111.111..

c) Belt Driven Slide:

Servo drive with a 1000 line encoder,
reduction gear 1:10, transport belt pitch 3 mm,
driving pulley 35 teeth, length unit 1 inch.

4*1000 pulses for one motor revolution,
40'000 pulses for one pulley revolution or for 105 mm,
105 mm = 105/25.4 = 4.133858 inches
40'000/4.133858 = 9676.19 pulses per inch,
SCALEK = 9676.19

3.3.2 Frequency Division Ratio, DIV

The Figure 3-3 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 = 11906/fmax [kHz]

With a servo drive, fmax is 4 times the maximum encoder frequency.
With DIV = 120, a fmax of approx. 100 kHz is obtained.

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.

Example:

500 line encoder, max, speed 1500 rev/min

1500 rev/min = 25 rev/sec,
fmax = 25*500*4 = 50'000 Hz,

DIV = 11906/50 = 238.

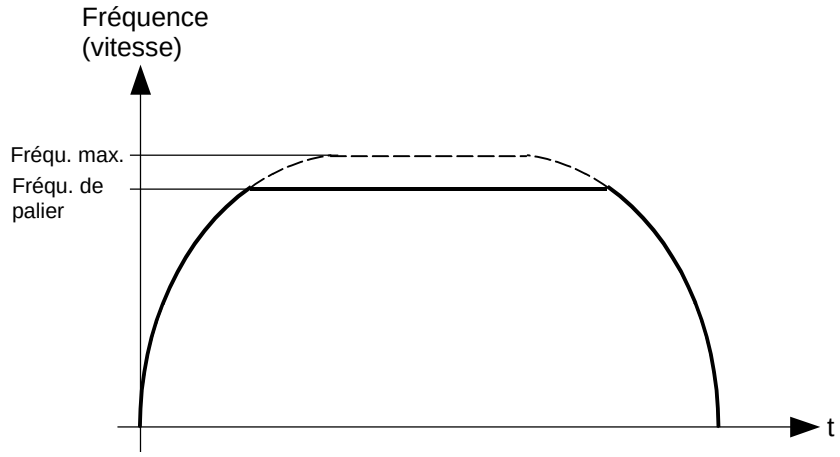


Figure 3-3 : Frequency or Speed versus Time

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/s2. As before, the frequency is the step frequency or 4 times the encoder 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]

For servo drives:
FEEDK = 4 * encoder frequency for one feed unit [kHz].

Examples:
(For the Electro-mechanical arrangements, see the examples of sect. 3.3.1)

1. 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

2. 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

3. Speed in yard/min.
   1 yard = 36 inches, 9676.19 * 36 for one yard,
   thus, 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

11906/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 exp 31 = 2.147 * 10 exp 9 .

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

3.3.6 Permanent Coordinate Translation, OFFSET

Please, refer to Figure 3-1. OFFSET must be entered in the length unit.

3.3.7 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-600-3 translators, BOOST is normally set to 1; at rest, the current is reduced to about 60% of it set value.

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

With the 5-phase translators E-600-1, BOOST may have several functions, according to the settings on the Berger translator board, see the E-600-1 Manual. BOOST = 0 has no action, the current has its nominal value.

The BOOST parameter is not applicable to servo drives.

3.4 Constants for Servo-Axes, Menu SERV.

The E-600 controller has two servo channels (position loops) designated CHANNEL 0 and CHANNEL 1. With the standard UNIPROG version, CHANNEL 0 corresponds to axis X and CHANNEL 1 to axis Y.

The function key F1 toggles the channel.

3.4.1 Servo Enable SERVO ENB.

SERVO ENB = 0: the X (or Y)-axis is a stepper axis
SERVO ENB = 1: the axis is a servo axis.

3.4.2 "In Position Zone", POS ZONE.

The axis is said to be in position when the magnitude of the position error is smaller than POS ZONE. A move is terminated only if this condition is met.

Valid range: 0..32'767 [encoder pulses * 4]

3.4.3 Alarm ZONE, ALM ZONE.

Whenever the magnitude of the position error exceeds ALM ZONE, the motion is stopped and an error message is issued, see section 7.3.

Valid range: 0..32'767 [encoder pulses * 4]

3.4.4 Gain of the Position Loop HI GAIN, LOW GAIN

Numerical value of the position loop gain.
Two values are provided for use with two mode servo amplifiers. UNIPROG operates in the LOM GAIN mode, the default mode.

With the Servo Adapter E600-4, E600-7, the analogue error signal is given by the expression:

AERR = NERR * 10/500 * GAIN/2exp16

or

AERR = NERR * GAIN * 3.052 * 10exp-7

AERR: error signal in Volts
NERR: numerical value of the position error, encoder pulses * 4
GAIN: parameter value

For more information, refer to E-600-4/7/12 manual.

3.5 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.

The closure check is possible only with a home switch assigned to REF INPUT 0 to 7.

4 Keyboard Operating Mode and UNIPROG Utilities

This chapter describes the operation of the E-600 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.

For the key designations, please refer to the diagram attached to this manual.

4.1 Switching the Power On.

The display shows the version of the system programs. Depressing the F2 key immediately after switching the power on may hold this screen.

```
* UNIPROG+ V x.xx *
  INT x.xx  NEC x.xx
```

After loading the program UNIPROG+ the controller E-600 displays:

```
RAM ERROR
FORMATED 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.

UNIPROG+ then makes an attempt to read the card plugged in the slot. Two situations are possible:

1. **The card slot is empty (or it contains an unformatted card):**
   The following message is displayed during a few seconds

```
LOADING BEE CARD
0.00,  WAIT
```

   Then:

```
CARD NOT FORMATTED
press any key
```

At this stage, the operator knows that the card has not been dumped into the RAM, thus, the previous contents of the RAM will be used for the following operation of the controller.

The depression of any key starts the "Power-Up" program, see chapter 7. The UNIPROG menus are now accessible.

Menu #1:

2. A valid card is plugged:

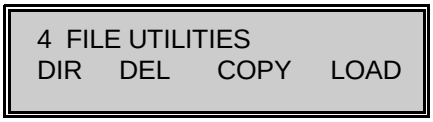
During the card dump, the screen shows:

x.xx is the label of the card, which was given while saving the data. With a valid card, the power-up program starts automatically and the menu #1 is on the screen.

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

4.2 Menu Selection

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



A menu #0 can be added on special request. This menu is intended to enter application specific parameters.

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.

```
NO ACCESS
press any key
```

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 F+ 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, Z, U) 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
OFFSET	Translation Home Position to Basis Coordinate System
BOOST	Current Booster Action , see the driver booklets

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.

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 3.4., enter engineering units
REF. INPUT	Assign one of the 8 inputs, LS(0) to LS(7) as the home switch input to the axis. Enter 8 for an axis without home switch and from 9 to 17 for a home position defined by a switch on the slide and an index on the motor shaft. see also sect. ....
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.

4.4.3 CTRL, Assignment of the Control Inputs

Enter the CTRL menu 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 push buttons and keys, see also section 5.3. If 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.

New parameters must be configured before using UNIPROG+ program.

DISPLAY FORMAT 1-6.	This sub-menu also contains the definition of the display format; input the number of digits to be written at right of the decimal point.
2 HANDS START	This parameter makes it possible to configure the start with two hands on 2 inputs chosen under EXTERNAL START and 2 HANDS START. (Value from 0 to 8). • In this case, only inputs from 0 to 7 are usable for these two parameters. • A value of 8 introduced into the parameter 2 HANDS START restores a simple start.

MANUAL PROGRAMME	This parameter makes it possible to associate the start of a program with a key from the numerical keypad. This must be done using the jogging menu (once the cursor disappears). (Value from 0 to 100).
LAST TOOL NB	This parameter (last number of the tool) makes it possible to limit the number of tools used, so as not to unnecessarily fill up program memory. (Value from 0 to 63).
MAX RPM 10 volts	This parameter makes it possible to configure the speed of the spindle with a value of 10 volts for the frequency converter. (Value from 0 to 99999).
LAST DELAYED OUT	This parameter makes it possible to configure a number of outputs for which the reset to zero is delayed. (Value from 0 to 7). • 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 SET BRK_D)
TOOL L/R INVERT	This parameter makes it possible to restore correspondence between the start direction of the tool at the time of the contouring and the correction of the trajectory. By entering the value "0" the direction of this axis is considered to be normal, and the reference of the surface is positive towards the upper right quadrant. (Value from 0 to 1).
LANGUAGE	The parameter makes it possible to display certain messages in the following languages: 0= English, 1= French, 2= German. (Value from 0 to 2).

4.4.4 SERV: Parameters to the Servo Driven Channels.

Starting at the CONFIGURATION menu, the F1 key introduces the sub-menu SERV. The parameters to be inputted are again organized as a two dimensional array. The arrow keys select the parameters and the F1 key toggles the channels.

- The servo channel 0, when used, is the X-axis, the channel 1 becomes axis Y.
- All servo parameter have been discussed in chapter 3.

SERVO ENB	The axis X or Y is used with a servo drive
POS ZONE	Position Zone, absolute value
ALM ZONE	Alarm Zone, absolute value
HI GAIN	Position loop gain, used with special amplifier only
LOW GAIN	Position loop gain

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 Table), JOG (Jogging) and CLOS (Closure check) are not available while a program is running; an attempt to enter one of this function is signaled by the message:

4.5.1 TOOL table.

In the menu "MOTION CONTROL," the submenu REF is replaced by TOOL. In this submenu the origins and the tool diameters are accessible. 4 origins and a diameter are associated with each tool to achieve correction of the tool trajectory.

The values, visible in this tool table, are stored in file " 0 " which is automatically opened when the parameter " LAST TOOL NB " is set. If by means of a false manipulation, file " 0 " does not have a suitable size the following message appears:

```
FILE 0 NOT FORMATTED
press any key
```

In this case, the parameter " LAST TOOL NB " must be reconfigured.

4.5.2 JOG, Jogging Motions

In this sub-menu, the upper display line shows the selected axis, its position in the basis coordinate system; the lower line displays the value of the incremental move effected by keys JOG+ or JOG-.

The REF key is also active as in section 4.5.1.

- **JOG+ and JOG- Keys:**

  Depressing one of these 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.

- **F3 and F4 Keys:**

  These keys select the value of "INCREMENT", F3 increases the value ten times, F4 decreases the value.

- **Arrow-Keys:**

  The arrow-keys 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 in the upper line, a destination coordinate -measured in the basis coordinate system- 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.

- **Special Function: Rotary Table, F1 Key**
  This utility is a complete division program for a rotary table. The axis is selected by the X, Y, ...keys and the scale factor is supposed to allow programming the angles in degrees.
  Screen:

```
DIVIDER            xx
CUR. DIVISION      yy
```

  With the arrow keys to move the cursor and the numerical pad, enter the number of regular divisions (xx) and the current division (yy). Generally, the current division will be 0 at the beginning. After this initialization, the JOG+ and JOG- keys will effect a rotation of exactly 360/xx degrees in positive or negative direction. If xx is negative, the direction of rotation is inverted.

  The ESC key does not return to the base menu but merely to the JOG sub-menu. Thus, other axes may be moved at each division of the table. Depress F1 to effect the next division.

- **Tool Origin definition**
  This utility allows, in addition to its function of the manual displacement, to record the origins for 64 different tools (from 0 to 63). The number of the tools used is configured by the parameter " LAST TOOL NB ".
  - The selection of the tool number is carried out by using keys " F1 and F2 "
  - The new origins are stored in file 0, which is automatically opened (this file contains FDATA ").
  - To easily consult these origins the tool table named " TOOL " is accessible from the menu 'MOTION CONTROL '
  - These new origins will be simply interpreted by instruction 47 TOOLP if a contour file is associated, or instruction 19 TOOL.
  - Only one value follows the instruction TOOL (tool number). Two values follow the instruction TOOLP (tool number and number of the contour file).
  - The key " PROG LINE " enters into the editor at the precise point of the previous exit.
  - During the program it is possible to stay in the JOGGING menu, but the movements are then locked.
  - The numerical keys from 0 to 9 start a program edited in UNIPROG+.
  - The attribution of the program to a key is defined by the parameter MANUAL PROGRAMS in the configuration CTRL. The number of the executed program is equal to the number in the parameter plus the numerical key.

    For example:
    MANUAL PROGRAM = 90
    Key 0 activates program 90,
    Key 1 activates program 91, etc.

    If the configured value exceeds 90 there is no longer a manual program. These manual functions are only activated in the jogging menu (JOG) and when the red LED for STOP and green LED for START are off.

- **Displacement Increment modification**
  To modify the increment of the displacement or to create an absolute displacement by jogging it is necessary to recall the blinking cursor by means of the arrow keys (low and high) and to place it over the value to be modified.
  - When the cursor disappears, the manual functions are activated.
  - By default the cursor is in the incremental mode.
  - The key " - " is equivalent to the key " 0 ".

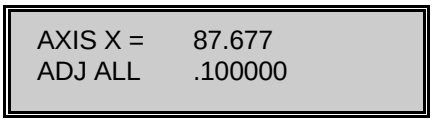
- **Recording a position is carried out in the following way:**
  1) Reach the position by jogging.
  2) Select tool " F1 " or " F2 " and the relevant axis.
  3) Press key " F5 " to authorize the input of the value of the new origin on the axis and the selected tool.
  4) Enter the value on the numerical keypad, then validate it by hitting the "Enter" key.

  Note:

  For small adjustments, the correction of the origin of the tool can be entered in an incremental way with the keys " + " and " - " in the jogging. The value, displayed in the lower right angle (the increment), will be removed or added to the origin. This is done without generating any displacement.

- **Shifting all of the tools:**
  In the lower left quadrant the display indicates " ALL ". If F1 is pressed the correction of the

origin will be applied to all of the tools, The selection is confirmed by the blinking LED F1 and the text at the bottom-left " ADJ ALL " (adjust all).



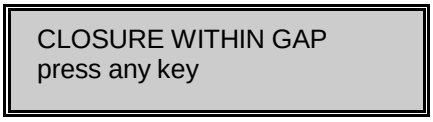
In such cases each hit on the key " + " in the jogging shifts the origin of all the tools while adding 0.1 mm and reciprocally with the " - " key. "F1" or "F2" can modify the increment value.

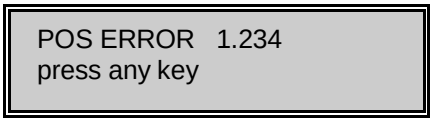
The modification of the coordinates of the origin can also be done from the table " TOOL ".

4.5.3 CLOS: Closure Check

It may be useful to effect a closure check, as described in section 3.4., to check the accuracy of the mechanical system. Depress F3 to enter the sub-menu, select the axis and depress REF.

The result of the check is displayed on the screen:





If the axis has no home switch or if it uses the index pulse of an encoder for the determination of the home position, 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 DISP: Axis Position Display

The DISP sub-menu (F3 key) displays the position of all 4 axes in the current coordinate system. DISP 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.

4.6 Menu "PROGRAMMING".

This menu contains all the functions needed to write, execute and to save programs. 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 arguments 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 onto the BEE CARD.

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

In order to avoid writing over valuable data, the warning

```
WANT TO SAVE B-CARD
  2.23  ?  YES  NO
```

will be displayed. The identification code (here 2.23) is the actual code of the card in the slot. If the operator wants to write the data without changing the code, it presses F3 and the saving starts. To change the code of the card or to abort the saving processes the operator answers "NO" by depressing F4.

```
WANT  TO  CHANGE
NAME   ?        NO
```

Now, F4 aborts the saving. Entering a code -or NAME- will start the writing of the card after the depression of ENTER.

```
SUCCESSFUL WRITING
press any key
```

or

```
WRITE ERROR
press any key
```

A writing error is an indication of an incorrectly plugged card, a missing card or even a defective card.

4.7 FILE UTILITIES: File Manipulation

The file utilities always act on the RAM contents; in order to alter a file on the Data BEE-CARD, first load the card into the RAM as despicted in section 4.7.4.

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 actions 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 Entire BEE-CARD into the RAM

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

```
WANT TO LOAD B-CARD
  1.03  ?  YES  NO
```

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

```
SUCCESSFUL LOADING
press any key
```

or

```
CARD NOT FORMATTED
press any key
```

The last screen indicates a missing card or a wrongly formatted card.

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.

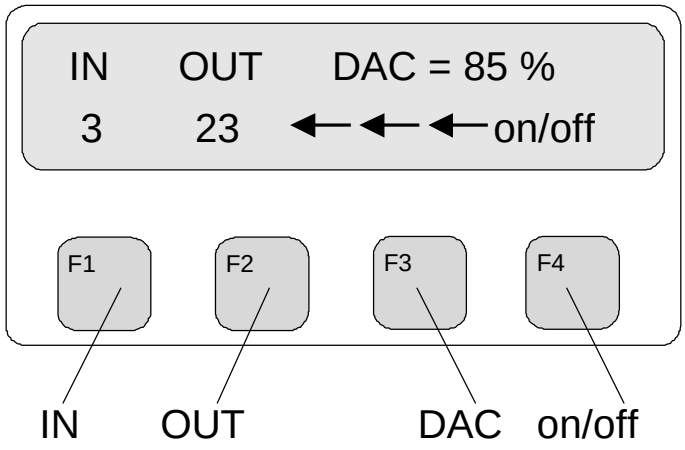


Figure 4-1 : "I/O Control" Menu

In the example above, the status of the input 3 is represented by the LED F1, the status of the output 23 by the LED F2 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 5.3., 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.

4.8.3 PRT: Print Utility

A printer or a PC-compatible computer may be connected through the RS 232 link to obtain a printout of the UNIPROG files or programs. Enter the number of the file to be printed and the printing start upon the depression of the ENTER key.

The printout format is as follows:

- Line number
- File number
- Numerical code of the instruction
- Instruction mnemonics and arguments

The serial port of the printer or the PC must be initialized per:

9600 Bauds, 8 bits, 1 stop bit, no parity.

With a PC, use the DOS COPY utility, example:

COPY COM1: <file name>

4.8.4 SERVO: Servo Channel Utility

This utility is intended to set-up servo drives, especially to trim the servo parameters. For more specific information, refer to the manuals of the various servo-adapter modules.

The F1 key (CHAnnel) selects the servo channel.

When the F2 LED is lit, the position loop is closed, i.e. the servo channel is really a position servo. The upper line of the screen displays the position error (P-ERR) in length units.

- The F2 key (POSition Loop) toggles the servo mode from position to velocity control.
- The F3 key (ENaBle) toggles the Servo Enable Line to the power amplifier. The LED is lit when the power is on.
- The F4 key (HIGH) is used with "two mode" servo amplifiers. When the F4 LED is on, the Mode switching line is active and the proportional gain is taken from the parameter "HI GAIN". The default mode is "LOW GAIN".

In the velocity mode (position loop open, F2 off), the upper line of the screen shows the value of the velocity reference in Volts (Analogue OUT). A new value may be entered at any time.

It is also possible to generate a square wave velocity reference to test the response of the velocity loop:

- Press F4 and enter the duration of the half-period in the lower line,
- Enter the amplitude in the upper line.

Zero duration must be entered to return to a constant velocity reference. The F2 key switches back to the position mode, the CLR key resets the velocity reference.

This utility can be used while a program is running to check the position error. The F2 and F3 key are, of course, made inactive.

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, and 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 4.6. If speed = 7, the velocity is supposed to be in the accumulator.

"mode-e" argument: Defines the execution mode. 0 to 3 are legal entries.

0: the motion is accounted for but not yet executed

1: all accounted motions for the designated axis are execute

2: all accounted motions in all axes are executed

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 definition

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".

- 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 4 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 section 4.4.2.

5.2.2 Closure Check:

18	CLOS	<axis>	<speed>

The closure check function was explained in section 3.4. The "speed" argument has the same meaning as in the positioning instructions.

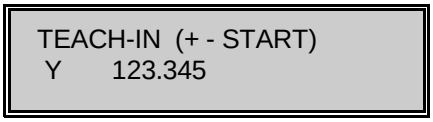
5.2.3 Teach-In Instruction:

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 chapter 7. It is mandatory to set the pause flag as explained in section 5.10.

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 these contents:

- An effective motion of the axis by the JOG keys,
- Entering corrections at the keypad.

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 Parameter definition

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

1	SET	PASSES	Value of the stripping passage.
2	SET	GAP	At the time of a quick return, assures beforehand a slow advance.
3	SET	DELAY	Basic time of the drilling.
4	SET	BRK-D	Time limit of the release of outputs 0 to 7 (LAST DELAYED OUTPUT).

The parameter numbers 0, 1, 2 relate to the drilling.

5.2.5 Peck cycle (drilling)

84 PECK <axis> <slow speed> <position end of drilling> <mode-d>

The drilling cycle determines the number of passes to create. The pass will always be a positive number. The position reached at the end of the drilling cycle is the absolute position in relation to the selected tool.

4 modes are possible:

Mode-d=0	Stripping with return at the end of the cycle to the starting position.
Mode-d=1	Breaking chips with return at the end of the cycle.
Mode-d=2	Stripping without return at the end of the cycle to the starting position.
Mode-d=3	Breaking chips without return at the end of the cycle.

- The quick return stops outside of machining area. The value of the space is called guard (GAP) and it is configured by the instruction SET 1 GAP with a default value of 0.1 mm.
- A delay configured by the instruction SET 2 DELAY is active at the end of the drilling in mode 0 and 1. The delay has a default value of 0.1 S.
- Modes 2 and 3 make it possible to chain together several drillings to increase progress.

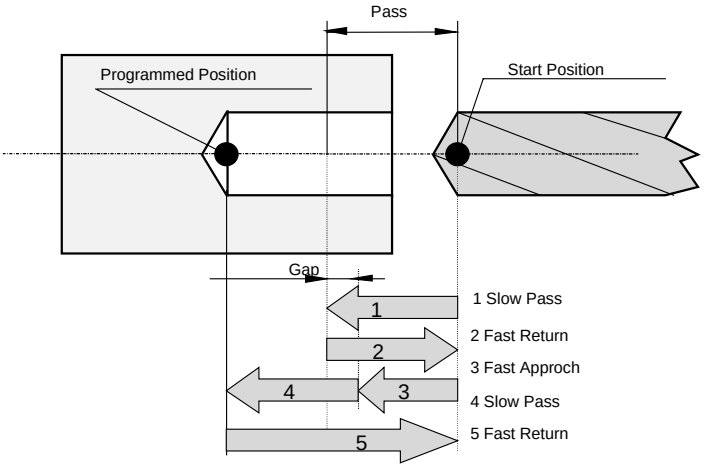


Figure 5-1 : Peck cycle

5.2.6 Tapping instruction

81 TPING <axis> <step> <final tapping position>

This instruction helps determine at which speed the machine advances by setting the spindle rotation speed and thread measurements.

- The instruction SPVEL determines the spindle rotation speed controlled by a frequency converter.
- The thread is selected in the menu FEED.
- To reverse the rotation at the end of the tapping, output 7 is reversed. It can therefore be cabled on the input "direction" of the frequency converter.
- The rotation of the spindle is reversed when the final position of the advance axis is reached. The rotation of the spindle is not measured and therefore does not affect the duration of the tapping. Consequently, the depth of the tapping can vary in the reverse way of the couple of tapping.

For example:
Tapping step of 1 mm; Rotation 600 rpm; depth 50

Menu Feed: RATE # 3 = 1.00 = Pitch

```
00 30   10   POSA    X 4 0.00 1
01 30   57   SPVEL   600          ; Disk speed
02 30   29   ON      7            ; Imposes the output direction 22 to 1
03 30   29   ON      6            ; Starts the converter
04 30   72   TPING   X 3 50       ; Starts the tapping
```

Note:
The speed at which the machine advances is determined without regard to the rotation speed of the spindle, which requires the use of a port-tool with **length compensation** to absorb the synchronization variations.

5.2.7 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.

If mode 0 is selected, the angle is recorded while waiting for the execution of the movement with the instruction RADIUS. In mode 1 and 2 the displacement on each axis is created each one at

their respective speed, in mode 3 the displacement is linear up to the next point.

5.2.8 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.
If mode 0 is selected, the radius is recorded while waiting for the execution of the movement with the instruction ANGLE. In mode 1 and 2 the displacement on each axes is created each one at their respective speed, in mode 3 the displacement is linear up to the next point.

5.2.9 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.

5.3 Input/Output 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>**
29	**ON**	**<output>**

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

The items of Table 5-1are recognized as input or output by UNIPROG.

Number in Instruction	Physical Input	Number in Instruction	Physical Output

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	SIM0	8	SIM0
9	SIM1	9	SIM1
10	SIM2	10	SIM2
11..15	FLAG(1..5)	11..15	FLAG(1..5)
16..63	IN(16..63)	16..63	OUT(16..63)

Table 5-1 : UNIPROG 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(16..63) 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 5.4. 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
C O O O	64...71
C C C C	0...7

Table 5-2 : IN/OUT Module Adresses

OUT(0) to OUT(7) are outputs implemented within the basic E-600 housing. They are available through the back panel connector, see chapter 9. They are not available with the "No Drive" version, E-600 ND, see chapter 10.

5.3.4 Complement of Output

95 CPL <output number>

This instruction reverses the output as an argument.

For example:
To activate and deactivate (toggle) an output 18 in a manual program 90 (key 0).

90 00	95	CPL	18
90 01	90	NOP	

5.4 Motor instructions

5.4.1 Spindle velocity

57	**SPVEL <rotations per minute>**

The parameter determines the speed in rotations per minute. The parameter MAX RPM 10 volts must be configured (with the value 100 the rotation speed is in percentages).

5.4.2 Frequency converter control

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

This instruction carries out the change in spindle motors by ensuring necessary time limits in order to change the frequency converter contractors.

The frequency converter must absolutely be changed without current and therefore at a speed of zero.

The instruction cycle MOTOR sets the speed to zero and then waits for the time limit set in the instruction SET 4 BRK-D (BReaK-Delay), before reactivating the speed directly defined by rotations per minute.

With " MOTOR 0 " all motors are set to zero.
With " MOTOR 1 " motor 2 is set to zero and motor 1 starts (OUT 1).
With "MOTOR 2 " motor 1 is set to zero and motor 2 starts (OUT 2).
etc... until MOTOR 7

5.5 Number Handling

UNIPROG works either with real (or floating) numbers or 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.

5.5.1 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>**

5.5.2 Store Accumulator

Store Accumulator Direct:

55	**STORD**	**<address>**

Store Accumulator Indirect:

56	**STORI**	**<pointer address>**

5.5.3 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.

5.5.4 Save a Variable into the Data Card:

54	**SAVE**	**<address>**

The variable in line "address" is saved in the Data Card (EEPROM) at the same address.

The write operation to the EEPROM may last for several tenths of a second, therefore this instruction is not recommended in portions of programs where timing is critical.

5.6 Program Control Instructions

5.6.1 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.

5.6.2 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.

5.6.3 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.

5.6.4 Repeat Loop:

The instructions in this section are intended to build 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
65	ENDRP		end of the loop

5.6.5 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 an 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 aborts all active tasks.

5.6.6 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 5.6.

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

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

5.7 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.

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

5.9 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.

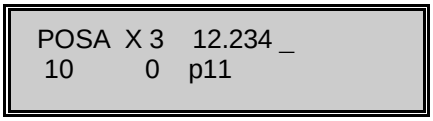
It is worthwhile to notice that the directives 98 and 99 act as NOP when written in a program. Thus, it possible to declare numerical variables anywhere in a program.

5.10 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 4.2. The screen prompts the operator to enter the file number to be edited. The content 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: ↑ goes to the previous line, ↓ goes to the next line.

PROG/LINE 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 to be edited 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 amount 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 is 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 F% 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 notices 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 reposition 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 belongs to the class 1o. Enter 10 (ENT key not required). The POSA symbol is displayed in the upper line, the cursor points to the argument "axis".
- Move the cursor to the left with the CLR key, then, depress F5 till the mnemonics POSR appears, now, press ENTER.
- The display now prompts the operator to select the axis, the selection can be done 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.

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 content 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 Program execution

The program execution is governed by the push buttons START, STOP and PAUSE, by the inputs designated in the CTRL configuration and by the execution mode selected by the MOD1 and MOD2 keys.

7.1 The Execution Modes, keys MOD1, MOD2

MOD1:
Normal execution mode. The Pause Flags in the instruction are ignored. The START button is lit.

MOD2:
The Pause Flag 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 push button is blinking. A depression of START restarts the program till the next flagged instruction. This mode is especially useful with the TRACE utility.

Execution mode 2 (blinking LED) limits the displacement speed while respecting the pre-determined stop points insert with the F1 key in edit mode. While in this mode and during the calculation procedure of the contour, the radii are indicated with respect to the order of execution of the contour. By hitting the blinking START key, the calculation continues.

7.2 START, PAUSE, STOP Key Functions

Remember that the inputs designated by the CTRL configuration are effectively ORed with this pushes 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

The MAN key pauses the running program at the end of the current instruction. However, a motion is immediately stopped to 0-velocity by the normal ramp-down process, i.e. the true position is preserved.

The START button is blinking and the key MAN lit. 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.

Notice:

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

Three 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).
3. The "alarm zone" of the position error of a servo-drive.

When 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**	**AXIS a ALARM ZONE** **press STOP**

Where "a" stands for the axis, X, Y, Z, U.

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-600 Series Controllers are able to generate vectors in all spaces with up to 4 dimensions. Contouring paths are obtained through the generation of small straight segments. The basic PINX-E language allows the programming of paths of any shape in cartesian, polar or cylindrical coordinate systems.

UNIPROG has a set of instruction to generate straight, circular and helical paths 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.

UNIPROG is able to generate vectors and paths in 11 different spaces:

0	1	2	3	4	5	6	7	8	9	10
XY	XZ	XU	YZ	YU	ZU	XYZ	XYU	XZU	YZU	XYZU

8.2 Vector Generation

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.

40 ORGP <axis> <absolute position>

The instruction ORGP (ORGin Path) 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.

8.3 Programming the Geometry of a Continuous Path

A continuous path is a concatenation of straight, circular and helical 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 paths. The limited computing power of the E-600 controller is overcome by a pre-interpretation of the geometry. The programmer has to instruct the path interpretation as depicted 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 paths 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>**
33	**LINR**	**<axis>**	**<component>**	**<mode>**

The pseudo-instructions LINA and LINR behave similarly to the instructions POSA and POSR. The concepts of the absolute coordinate and relative displacement refer to the space of the contour, 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
.....
.....
```

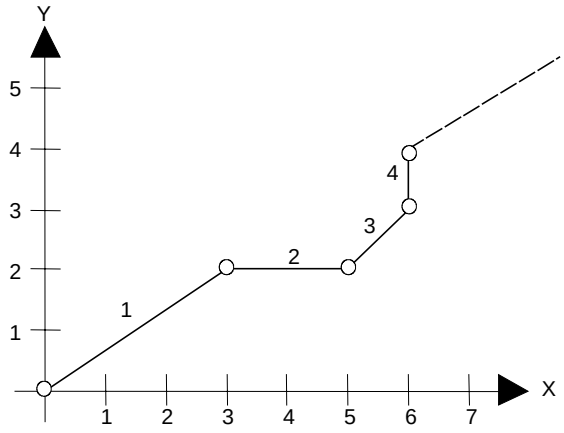


Figure 8-1 : Contour with Zero diameter

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>

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.

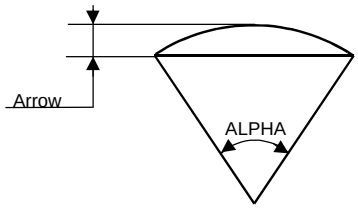
In a contour defined by several POINT instructions, it is not necessary to recall the last coordinate if it is identical.

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

35 CDEF <maximum arrow>

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>

The mode of the instruction RAD, 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, Figure 8-2.

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.

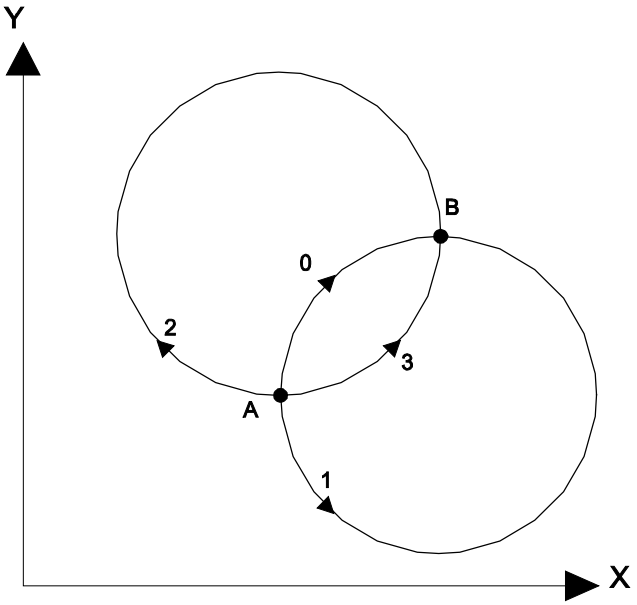


Figure 8-2 : Rotation modes (mode-r)

36	**CIRA**	**<axis>**	**<coordinate>**	**<mode>**
37	**CIRR**	**<axis>**	**<component>**	**<mode**

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.

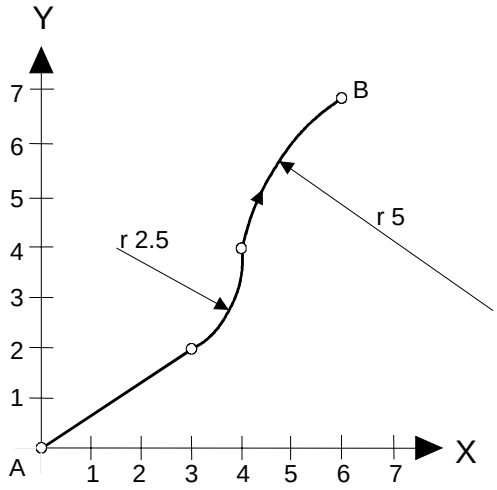


Figure 8-3 : Contour example

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

```
DPATH   XY      L
LINA    X       3 0
LINA    Y       2 2
RAD     3       2.5
CIRA    X       4 0
CIRA    Y       4 2
RAD     0       5
CIRA    X       6 0
CIRA    Y       7 2
END
```

```
DPATH   XY      L
LINR    X       3 0
LINR    Y       2 2
RAD     3       2.5     ;optional
CIRR    X       1 0
CIRR    Y       2 2
RAD     0       5       ;optional
CIRR    X       2 0
CIRR    Y       3 2
END
```

8.3.3 Helical Segment

A circular segment in a 3 or 4-dimensional space becomes a helical (hyper-helical) segment. The axes outside the circle effect linear motions, which start and stop simultaneously with the circular motion. A possible application is an helical displacement in a rectangular axis system while holding a tool normal to the path.

The pseudo-instruction LINA/LINR must be written after the CIRA/CIRR pseudo-instructions.

Example: Coding for a hyper-helical path:

```
DPATH   XYZU
CDEF    0.1                 ; Max. Arrow = 0,1mm
RAD     0       12.0        ; radius 12 mm, mode 0
CIRA    X       24.0  0
CIRA    Y       0   0       ; half circle
LINA    Z       15.0  0     ; helical pitch 30 mm
LINA    U       0.5   2     ; axis U moves 0.5 units
.....
```

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 paths 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 Length Limitations in the Vector Generation

The length S of a vector generated by a positioning instruction with mode 3, expressed in engineering units has to meet

$$S < 67'000'000 / KMUL_{max}$$

Where $KMUL_{max}$ is the largest value of the product [SCALEK(axis) * DIV(axis)] over all axes involved in the vector.

Example:
Space: DPATH XY

SKALEK(X)	= 1000 steps/mm
DIV(X)	= 300
KMUL(X)	= 300'000
SKALEK(Y)	= 1000 steps/mm
DIV(Y)	= 200
KMUL(Y)	= 200'000
$KMUL_{max}$	= 300'000

S < 223 mm

8.6 Execution of a Path

48 PATH <speed >

This instruction was previously instruction 38. The parameter, file number, disappears, the file is 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.

46 CORR <speed >

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

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.

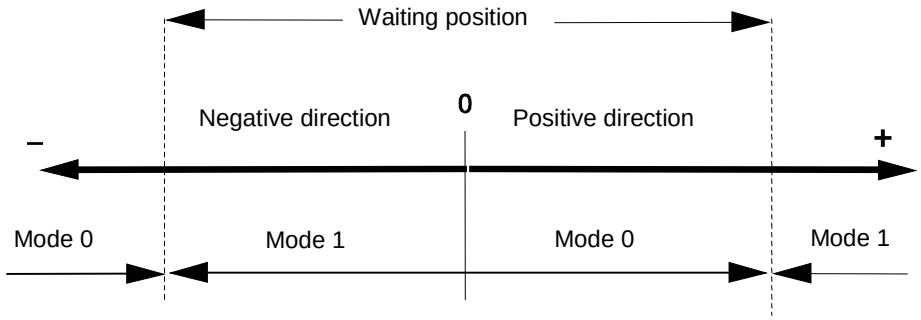
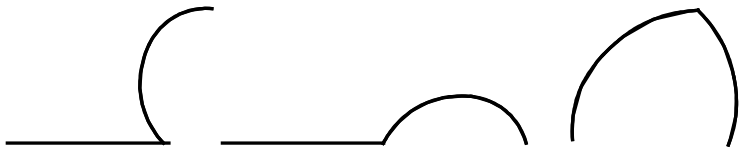


Figure 8-4 : Waiting position according to the displacement direction

8.7 Case not accepting the correction of the tool

UNIPROG+ does not solve all cases of contouring, 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.



8.8 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.

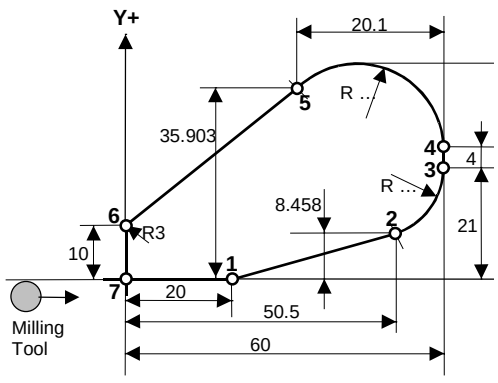
RAYON < 1: The contour presents at least a radius smaller than 1 mm. Consequently the execution speed of the contour will have to be adapted to run the radius.

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.

8.9 Contouring example

" TOOL " tables

ORIGIN X	TOOL 0	?
ORIGIN Y	TOOL 0	?
ORIGIN Z	TOOL 0	?
ORIGIN U	TOOL 0	?
DIAMETRE	TOOL 0	?
ORIGIN X	TOOL 1	13
ORIGIN Y	TOOL 1	54
ORIGIN Z	TOOL 1	?
ORIGIN U	TOOL 1	?
DIAMETRE	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  -20.100  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
```

8.10 Summary of Contouring Instructions and Pseudo-Instructions

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description	Page
35	**CDEF**	Max. arrow				Define segmentation value	45
36	**CIRA**	Axis	Coordinate	Mode-e		Define absolute circular mvt	46
37	**CIRR**	Axis	Component	Mode-e		Define relative circular mvt	46
46	**CORR**	Speed				Rectilinear displacement	48
30	**DPATH**	Space	Tool L-R			Define working Plan	43
66	**ENDP**					End of path	48
32	**LINA**	Axis	Coordinate	Mode-e		Define absolute straight segm.	44
33	**LINR**	Axis	Component	Mode-e		Define relative straight segm.	44
40	**ORGP**	Axis	Abs. Pos.			Set Path origin	43
48	**PATH**	Speed				Path execution	48
31	**PCOMP**	File number				Geometric file interpretation	47
42	**POINT**	Axis	Angle pos.	Mode-e		Segm. bounded with rounding	45
34	**RAD**	Mode-r	Radius			Define radius and rotation dir.	45
47	**TOOLP**	Tool Nr	File Nr			Geometric file interpretation	47
82	**WAITP**	Axis	Speed	position	Mode-w	Waiting reaching a position	48

9 UNIPROG+ Recapitulation

9.1 Instructions

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description	Page
91	**ADDD**	address				Add to Accu Direct	38
86	**ANGLE**	Speed	Value	Mode-e		Displ. angle (polar coordinate)	32
22	**BRIN0**	input	address			Branch if Input is False	33
23	**BRIN1**	input	address			Branch if Input is True	33
24	**BRM**	address				Branch if Accu is <0	37
27	**BRNZ**	address				Branch if Accu is non zero	37
25	**BRP**	address				Branch if Accu is >=0	37
26	**BRZ**	address				Branch if Accu is zero	37
61	**CALL**	address				Sub-Routine Call	36
35	**CDEF**	Max. arrow				Define segmentation value	45
36	**CIRA**	Axis	Coordinate	Mode-e		Define absolute circular mvt	46
37	**CIRR**	Axis	Component	Mode-e		Define relative circular mvt	46
18	**CLOS**	axis	speed			Closure Check	30
46	**CORR**	Speed				Rectilinear displacement	48
95	**CPL**	Output Nr				Output state complement	34
59	**DECD**	address				Decrement Address Direct	36
94	**DIVD**	address				Divide Accu Direct	38
30	**DPATH**	Space	Tool L-R			Define working Plan	43
62	**END**					End of Prog. and Routines	36
66	**ENDP**					End of path	48
65	**ENDRP**					End of Repeat Loop	37
98	**FDATA**	floating				Define a Floating Number	38
50	**FLOAD**	number				Load Accum Immed.,Floating	35
99	**IDATA**	integer				Define an Integer Number	38
51	**ILOAD**	integer				Load Accum Immed.,Integer	35
58	**INCD**	address				Increment Address Direct	36
60	**JMP**	address				Unconditional Jump	36
32	**LINA**	Axis	Coordinate	Mode-e		Define absolute straight segm.	44
33	**LINR**	Axis	Component	Mode-e		Define relative straight segm.	44
52	**LOADD**	address				Load Accum Direct	35
53	**LOADI**	pointer				Load Accum Indirect	35
85	**MOTOR**	Motor Nr	Rot. Speed			Motor rotation speed	35
93	**MULD**	address				Multiply Accu Direct	38
90	**NOP**					No Operation	38
28	**OFF**	output				Set Output to 0	33
29	**ON**	output				Set Output to 1	33
88	**ORGA**	Angul. shift				Angular shift of reference axis	33
40	**ORGP**	Axis	Abs. Pos.			Set Path origin	43
48	**PATH**	Speed				Path execution	48
31	**PCOMP**	File number				Geometric file interpretation	47
84	**PECK**	Axis	Slow speed	Drill pos.	Mode-d	Peck cycle (drilling)	31
42	**POINT**	Axis	Angle pos.	Mode-e		Segm. bounded with rounding	45
10	**POSA**	axis	speed	coordinate	Mode-e	Absolute Indexing,Immed.	29

Code	Instruction	1st arg.	2nd arg.	3rd arg.	4th arg.	Description	Page
11	**POSAD**	axis	speed	address	Mode-e	Absolute Indexing,Direct	29
12	**POSAI**	axis	speed	pointer	Mode-e	Absolute Index. Indirect	29
14	**POSR**	axis	speed	disp'ent	Mode-e	Relative Indexing, Immed.	29
15	**POSRD**	axis	speed	address	Mode-e	Relative Indexing, Direct	29
16	**POSRI**	axis	speed	pointer	Mode-e	Relative Index. Indirect	30
34	**RAD**	Mode-r	Radius			Define radius and rotation dir.	45
87	**RADIUS**	Speed	Value	Mode-e		Displ'mt. radius (polar coordinate)	33
17	**REF**	axis				Reference Point	30
63	**REP**	n times				Repeat, Immediate n	37
64	**REPD**	address				Repeat, Direct n	37
54	**SAVE**	address				Save to EEPROM	36
83	**SET**	Param. Nr	Param. Val.			Setting of variables	31
67	**SIM1**	address				1st Simultan. Prog. Call	37
68	**SIM2**	address				2nd Simultan. Prog. Call	37
57	**SPVEL**	Rot/min				Spindle speed	35
55	**STORD**	address				Store Accum Direct	36
56	**STORI**	pointer				Store Accum Indirect	36
92	**SUBD**	address				Subtract from Accu Direct	38
13	**TEACH**	axis	speed	address		Teach-In,Direct Argument	30
19	**TOOL**	Tool numb.				Set Tool parameters	30
47	**TOOLP**	Tool Nr	File Nr			Geometric file interpretation	47
81	**TPING**	Axis	Pitch	Tap. Pos.		Tapping instruction	32
70	**WAIT**	time				Dead Timer, Immed. Time	37
20	**WAIT0**	input				Wait if Input is False	33
21	**WAIT1**	input				Wait if Input is True	33
71	**WAITD**	address				Dead Timer, Direct Time	37
82	**WAITP**	Axis	Speed	position	Mode-w	Waiting reaching a position	48

9.2 Inputs - 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	SIM0	8	SIM0
9	SIM1	9	SIM1
10	SIM2	0	SIM2
11	FLAG(1)	11	FLAG(1)
12	FLAG(2)	12	FLAG(2)
13	FLAG(3)	13	FLAG(3)
14	FLAG(4)	14	FLAG(4)
15	FLAG(5)	15	FLAG(5)
16..63	IN(16..63)	16..63	OUT(16..63)

9.3 Divers

Axis:	X = 0	Y = 1		Z = 2	U = 3						
Channel:	X = 0	Y = 1		Key F1							
Spaces:	0	1	2	3	4	5	6	7	8	9	10
	XY	XZ	XU	YZ	YU	ZU	XYZ	XYU	XZU	YZU	XYZU

Pause Flag: Key and Led F1 **Insert**: Key F3 **Delete**: Key F4