UNIDEX® 21
PROGRAMMING MANUAL

PN: EDU118

AEROTECH, Inc., 101 Zeta Drive, Pittsburgh, PA 15238
(412) 963-7470 • TWX 710-795-3125 • FAX(412) 963-7459

DISCLAIMER:

The information contained in this manual is subject to change due to improvements in design. Though this document has been checked for inaccuracies, Aerotech does not assume responsibility for any errors contained herein.

TRADEMARKS:

Unidex is a registered trademark of Aerotech, Inc.

IBM PC/XT/AT are registered trademarks of the International Business Machines Corporation.

PAMUX is a registered trademark of the Opto 22 Corporation.

REVISION HISTORY:

Revision 1 - February, 1992
Revision 2 - January, 1993

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

This manual provides detailed information for each of the program commands utilized by the Unidex 21 Motion Controller.

SECTION 1-1: COMMAND STRUCTURE

Commands recognized by the Unidex 21 consist of three general types:

1. A subset of RS-274-D commands (G & M codes)
2. A subset of RS-447 commands (i.e., DENT, DFS, etc.)
3. Aerotech developed extensions (i.e., $INP, $OTP, etc.)

The programming language syntax for these commands are of two types:

TYPE 1 : Machine program data in RS-274-D like format.

TYPE 2 : Machine setup, initialization, or operation parameters in RS-447 like format. Type 2 commands are designated by either ( ) or < >.

NOTE: The Type 1 (RS-274-D) format requires a space between each command (for instance G2 G17 I1).

Both Type 1 and Type 2 commands conform to the Electronic Industries Association (EIA) standards.

By conforming to these EIA standards wherever possible, the Unidex 21 program commands may be readily converted to/from other CNC controllers. Also, CAD to NC postprocessors may be easily configured.

SECTION 1-2: ENTERING PROGRAM COMMANDS

The Unidex 21 recognizes program commands created in the following ways:

The Unidex 21 Edit Mode

The Unidex 21 Motion Controller contains an Edit mode in which programs may be created or amended. The Front Panel or integral keyboard is used to enter or revise the commands. See the *Unidex 21 User's Manual* for a detailed description of the Edit Mode.

The Unidex 21 Machine Mode

The Machine Mode of the Unidex 21 may be used to amend program commands of a previously loaded program. The MDI function of the Machine Mode provides the User with the ability to stop the program, enter a move or change a status, and then return to the program run. See the *Unidex 21 User's Manual* for a detailed description of the Machine Mode.

IBM/AT Computer - DOS Operating System

Programs may be created remotely if an IBM/AT computer equipped with the IBM-DOS Operating System is used. Programs created in this manner may be copied onto a 3 1/2" floppy disk and then directly copied into the Unidex 21's memory. See the *Unidex 21 User's Manual* for a detailed description of the File Mode.

Non-IBM Computer

A non-IBM computer may be used to create programs if it is interfaced to the Unidex 21 through one of the RS-232 ports. See the *Unidex 21 Hardware Manual* for information concerning RS-232 interface and the *Unidex 21 User's Manual* for a detailed description the File Mode and the procedure of loading a file through the RS-232 ports.

CHAPTER 2: USING THE UNIDEX 21 PROGRAMMING MANUAL

This manual is to be used as an aid when programming the Unidex 21 Motion Controller. Prior to any actual programming of the Unidex 21, it is strongly suggested that the User be thoroughly familiar with the *Unidex 21 User's Manual* and have studied the Sample Programs provided throughout Chapter 3 and the appendices of this manual.

The Unidex 21's internal HELP menu is available to the User at all times by pressing the "Alt" and "H" keys.

The commands presented in this manual are in alphabetical order, symbol commands are listed first. Each command is described on an individual page with an example of it's usage. Names of related commands or commands whose description may be helpful are also included.

It should be noted that one of the most important programming capabilities which separates the Unidex 21 from other similar controllers is its extensive variable programming capability.

The following is a sample page with an explanation of the various subsections.

XX - Command

NAME:

The command name.

FUNCTION:

A description of the command and any parameters that are required by the command.

FORMAT:

A description of the command structure.

RETURNS:

Feedback to the User following execution of the command.

EXAMPLE:

```
An example of command use                    ; Explanation of the example (not intended for
                                               system entry).
```

PROGRAMMING EXAMPLE:

```
An example of command used in conjunction with other commands   ; Explanation of the com-
                                                                ; mand(s) in the example
                                                                ; (not intended for system
                                                                ; entry).
```

NOTES:

Additional comments pertaining to the use of the command.

RELATED COMMANDS:

The names of additional commands that are related or whose description may be helpful.

CHAPTER 3: UNIDEX 21 PROGRAM COMMANDS

The following pages contain the commands utilized by the Unidex 21.

The Unidex 21 utilizes a combination of standard EIA and RS274-D type commands (G & M code) and an extended subset of RS447 commands.

Each command is presented alphabetically with system level variables and operands first.

%

NAME:

% - Program Title

FUNCTION:

The % command may be used as an option for beginning a program. Any text located between the % and the first <ENTER> is ignored by the Unidex 21. A program name or a comment may also be inserted into this block. Programs which are transferred via the RS-232 and IEEE488 interface may utilize the "%" as a beginning of file transmission character. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode settings.)

FORMAT:

% Program Title <ENTER>

RETURNS:

EXAMPLE:

```
% Widget Cutting <ENTER>     ; The program name is Widget Cutting.
G1                           ; All moves will be linear.
G70                          ; Initiate English programming (inches).
F100.                        ; A Feedrate of 100 inches per minute is established.
X10. Y10.                    ; Move X axis 10 inches in the positive direction,
                             ; and Y axis 10 inches in the positive direction.
```

NOTES:

The applicable Main System Parameters for RS-232 and IEEE488 communication are #6, 8, 10, 16, 17, and 53 (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode).

RELATED COMMANDS:

;, /

/

NAME:

/ - Block Delete Operator

FUNCTION:

The Block Delete operator permits the User to designate program blocks to be skipped during a program run. During program editing, a "/" may be inserted at the block's beginning to designate an optional block. Pressing the front panel **BLOCK DELETE** key toggles between activating the Block Delete function (designated blocks are skipped) or deactivating the Block Delete (all blocks are executed regardless of designation). From an AT type keyboard the Prt Sc (print screen function key) will also toggle the Block Delete.

FORMAT:

/(command name)

RETURNS:

EXAMPLE:

```
(DFS,TEST
 .
 .
 (JUMP,ENT1,VAR1.EQ.2)      ; Jump to Entry Point ENT1 if VAR1 equals 2. The system
 /(ABTS,ENT2)               ; would not execute the ABTS command when BLOCK
 (DENT,ENT1)                ; DELETE is active.
 .
)
(DENT,ENT2)
```

This example controls a jump to the Entry Point ENT2 before completion of the "TEST" Subroutine. If **BLOCK DELETE** is OFF and VAR1 does not equal 2 then the Abort Subroutine **ABTS** command will be executed.

NOTES:

RELATED COMMANDS:

;

;

NAME:

; - Comment Operator

FUNCTION:

Comments may be added to any command by preceding them with the ;. Any text following the ; is ignored by the Unidex 21.

FORMAT:

(program command) ; comment

RETURNS:

EXAMPLE:

```
(REF,Y,y,Z,z)          ; Axes Y, y, Z, and z are at the Software Home position.

G1                     ; All moves will be linear.
G70                    ; Initiate English programming (inches).
F100                   ; A Feedrate of 100 inches per minute is established.
X10. Y5.               ; Move X axis 10 inches in the positive direction,
                       ; and Y axis 5 inches in the positive direction.
```

NOTES:

Each comment line must be preceded with the ";" Comment Operator.

RELATED COMMANDS:

%, /

$HSI

NAME:

$HSI - High Speed Interrupt Buffer

FUNCTION:

The **$HSI** command is a system variable that reads one element of data from the High Speed Interrupt Buffer. Data elements are floating point format.

FORMAT:

$HSI<0>	; Get number of data elements which have been stored.
$HSI<N>	; Get Nth data element which has been stored. Note that N may range from 1 to the number returned by **$HSI**<0>.

RETURNS:

EXAMPLE:

Assume High Speed Interrupt Buffer is active and contains the following:
0.0, 16.3, 22.4, 64.75, remainder not used

```
VAR1=$HSI<0>                ; VAR1= 4, the number of data elements stored.
VAR2=0                      ; Set Variable VAR2 equal to 0.
VAR3=1                      ; Initialize the data pointer.
(RPT, VAR1                  ; Repeat the Loop 4 times.
 VAR2=VAR2+ $HSI<VAR3>      ; VAR2 = 0.0, then 16.3, then 38.7, and finally 103.45.
 VAR3=VAR3 + 1              ; Increment the data pointer to 2, 3, 4, and 5,
                            ; note, however, that 5 is not used to read data.
)

VAR2 =VAR2/VAR1             ; Average data, VAR2 = 25.8625
```

NOTES:

For a more complete example utilizing the **$HSI** and related commands, see the example under the **HSIE** command

RELATED COMMANDS:

HSIE, MALC

$INn

NAME:

$INn - Input System Variable

FUNCTION:

The Unidex 21 has 16 optoisolated inputs available on the CPU card. The **$INn** represents the logic state of one of the input lines when n=0 to F. Or, a **$INP** command may be utilized to look at the logic state of all 16 input lines at once. These inputs typically interface to an I/O mounting rack. The values are recorded in hexadecimal and may be shifted, summed, or converted utilizing Unidex 21's Math Package. When programming the outputs, the User must recognize that input values are reversed relative to their module location. Therefore, if outputs 0 and 1 are set to zero active (ON) and all others on the I/O mounting rack are inactive (OFF), then the associated **$INP** hexadecimal status value is FFFC **not** CFFF.

FORMAT:

$IN0-$INF ; Represents individual input variable logic states.

or

$INP ; Represents all 16 input variable logic states. Used when you wish to control program flow based upon entire word.

RETURNS:

EXAMPLES:

```
VAR1=$IN0                          ; If Input 0 is active (ON) then VAR1=H,00.

(JUMP,CRY1,$INP.EQ.H,0007)         ; Jump to Entry Point CRY1 if inputs 0, 1, and
                                     2 are inactive (OFF) and all other outputs are
                                     active (ON).

(JUMP,CRY2,$IN1.EQ.H,00)           ; Jump if the value of Input equals H,00 active
                                     (ON).

(MSG,#H,$INP)                      ; Display value of inputs in hexadecimal; val-
                                     ues would range from 0 to FFFF. This is
                                     useful for debug purposes in order to check
                                     the value of inputs.
```

$INn

NOTES:

1) Valid individual input characters are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

2) On a standard I/O mounting rack assembly, PB16 or PB24 Input 0 (**$IN0**) actually resides at module position 8. (Modue positions 1 to 7 are reserved as outputs, see **$OTn**.)

3) Additional interrupt inputs, high speed position grabbing, and fast feedhold inputs are also available.

4) Reference the *Unidex 21 Hardware Manual* for further details.

RELATED COMMANDS:

$OTn, DVAR, HSIE, INT1/INT2, MSG, SIOC

$MFO

NAME:

$MFO - Manual Feed Override System Variable

FUNCTION:

Represents the current percentage of Manual Feedrate Override. Manual Feedrate Override is the multiplier used to raise or lower the Feedrate at which the system is operating. Or, a percentage multiplier of a programmed base. As an example, if a User's Parts Program specifies a Feedrate of 100.0 inches per minute a Manual Feedrate Override setting of 125% would cause the system to operate at 125.0 inches per minute.

FORMAT:

$MFO

$MFO may be equated to any other variable.

RETURNS:

Returned value is the percentage value of the current **MFO** divided by 100.

For example: $\$MFO = \frac{MFO\%}{100}$

EXAMPLES:

```
VAR1=$MFO          ; If MFO is set to 150%, VAR1 will equal 1.5.

(MSG,#$MFO)        ; The current MFO setting will be displayed as 1.25 if the MFO
                     is set at 125%.
```

NOTES:

1) While **$MFO** is the **MFO** percentage divided by 100, the **UMFO** program command is the actual % value. i.e., 125% equals 125.

2) **G0** moves are truncated at 100% maximum **MFO**.

RELATED COMMANDS:

CPAG, DVAR, MSG, UMFO

$nAP/$nRP

NAME:

$nAP - Absolute Commanded Position Register Variables

$nRP - Relative Commanded Position Register Variables

FUNCTION:

The Unidex 21 utilizes two position registers per axis referred to as Absolute and Relative Axis Commanded Position Registers. Each may contain the same or different value depending upon the interaction of other system commands like **FXOF**, **G92**, **HOME**, and **MORG**. The Tracking Display typically shows the **$nRP** values.

The **$nAP** refers to the axes Absolute Commanded Position Registers, where "n" is the primary axis letter (i.e., **X**, **Y**, **Z**, **U**, **x**, **y**, **z**, **u**). Upon homing the system or initial power up, these registers are initialized at zero. These registers are unaffected by the **G92** command, yet may be utilized to reinitialize the **$nRP** registers to **$nAP** values. This is especially useful if switching between **G90** and **G91** modes in which case the User is unsure of his exact position. They may also be utilized for verification of offset values placed in the User's Parts Program.

The **$nRP** refers to the axes Relative Commanded Position Registers, where "n" is the primary axis letter (i.e., **X**, **Y**, **Z**, **U**, **x**, **y**, **z**, **u**). Upon homing the system or initial power up, these registers are also initialized at zero. These registers may be changed by utilizing the **G92** command. When the control makes a move in **G90** or **G91** mode, it utilizes the **$nRP** registers. See Notes for special cases and related commands.

FORMAT:

$XAP $YAP $ZAP $UAP $xAP $yAP $zAP $uAP
$XRP $YRP $ZRP $URP $xRP $yRP $zRP $uRP

These expressions may be equated to any other variable.

RETURNS:

$nAP/$nRP

EXAMPLES:

```
VAR1=$xRP                    ; VAR1 is set equal to the Relative Position of the x axis.

VAR1=$xAP+$XAP               ; VAR1 is set equal to the sum of the Absolute Position of the
                               x axis and the X axis.

(MSG, X position is #$XAP)   ; Message display will show: "X position is 25.000" if the
                               current Absolute Position is 25.000.

G92 X=$XAP                   ; The system X axis Relative Machine Register Position is
                               reinitialized to the Absolute Position Register value. The
                               machine's position Tracking Display will also be changed.
```

PROGRAMMING EXAMPLE:

The following is an example of how the system variables relating to Absolute and Relative Positions are related to each other as well as how they are affected by the various commands.

NOTE: The Machine Origin steps parameter for the X axis corresponds with 10.0 inches. The Y axis parameter corresponds with 25.0 inches.

```
G1                           ; All moves will be linear.
G17                          ; The XY plane is the 1st plane.
G40                          ; Ensure Cutter Compensation is OFF.
G91                          ; Initiate Incremental Positioning.
G70                          ; Initiate English Programming (inches).
F1000.                       ; Feedrate is 1000 inches per minute.
(REF, X, Y)                  ; Find the Hardware Home for the X and Y axes.
                             ;    $XRP=0.0    $YRP=0.0
                             ;    $XAP=0.0    $YAP=0.0
X10. Y100.                   ; Move X axis 10 inches in the positive direction,
                             ; and Y axis 100 inches in the positive direction.
                             ;    $XRP=10.0   $YRP=100.0
                             ;    $XAP=10.0   $YAP=100.0
X10. Y-10.                   ; Move X axis 10 inches in the positive direction,
                             ; and Y axis 10 inches in the negative direction.
                             ;    $XRP=20.0   $YRP=90.0
                             ;    $XAP=20.0   $YAP=90.0
```

$nAP/$nRP

PROGRAMMING EXAMPLE (CON'T):

```
G92 X0. Y0.          ; Establish a Software Home at the current position.
                     ;    $XRP=0.0    $YRP=0.0
                     ;    $XAP=20.0   $YAP=90.0
X-20. Y35.           ; Move X axis 20 inches in the negative direction,
                     ; and Y axis 35 inches in the positive direction.
                     ;    $XRP=-20.0  $YRP=35.0
                     ;    $XAP=0.0    $YAP=125.0
(FXOF, X100., Y100.) ; Move axes to a new coordinate frame such that $nRP
                     ; registers retain their value while $nAP values are
                     ; changed.
                     ;    $XRP=-20.0  $YRP=35.0
                     ;    $XAP=100.0  $YAP=100.0
X10. Y15.            ; Move X axis 10 inches in the positive direction,
                     ; and Y axis 15 inches in the positive direction.
                     ;    $XRP=-10.0  $YRP=50.0
                     ;    $XAP=110.0  $YAP=115.0
(MORG, X,Y)          ; Move to the Machine Origin for these axes, which is
                     ; the Hardware Home position including Machine Origin
                     ; Offset while not disturbing the $nRP registers.
                     ; (See Note above)
                     ;    $XRP=-10.0  $YRP=50.0
                     ;    $XAP=10.0   $YAP=25.0
X20. Y50.            ; Move X axis 20 inches in the positive direction,
                     ; and Y axis 50 inches in the positive direction.
                     ;    $XRP=10.0   $YRP=100.0
                     ;    $XAP=30.0   $YAP=75.0
G90                  ; Initiate Absolute Positioning.
X100. Y100.          ; Move to Absolute Position 100., 100.  In this case,
                     ; the Y axis does not move.
                     ;    $XRP=100.0  $YRP=100.0
                     ;    $XAP=120.0  $YAP=75.0
M2                   ; End of The Program.
```

$nAP/$nRP

NOTES:

1) Normally upon power up or after homing, the **$nAP** and **$nRP** registers will equal the same value. The exception is when an Absolute Positioning system (i.e., resolver/inductosyn combination) is utilized. Here, the **$nAP** and **$nRP** registers will equal the machines Absolute Position.

2) The related commands, listed below, also affect **$nAP** and **$nRP** commands.

3) Usage of Machine Origin and Home Offset parameters also affect **$nAP** and **$nRP** commands.

4) The applicable Individual Axis Parameter is #7 as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

5) **$nAP** and **$nRP** reflect the commanded position of the axis and not necessarily the actual axis position.

RELATED COMMANDS:

FXOF, G92, HOME, MORG, REF

$OTn

NAME:

$OTn - Address Logic Output System Variable

FUNCTION:

The Unidex 21 has 8 optoisolated outputs available on the CPU card. The **$OTn** represents these outputs, where "n" is the output number ranging from 0 - 7. Alternatively, **$OTP** may be used to write all 8 output lines at once. The values are set up as hexadecimal, and may be shifted, summed, or converted utilizing Unidex 21's Math Package (see Chapter 4). When programming the outputs, the User must recognize that outputs are programmed in reverse to their module location. As an example, to turn ON outputs 0, 1, and 2 (set to zero) and all other outputs OFF (set to one) program **$OTP**=H, F8 **not** 8F.

These outputs typically interface to an I/O mounting rack. As opposed to MST bus outputs, if an output is commanded between motion blocks, motion will be momentarily interrupted during processing. If output on-the-fly is desired, utilize the MST bus instead.

FORMAT:

$OTn=x	; In this case, "n" is the Output number (0-7) and "x" equals 0 (ON) or 1 (OFF). An alternate way is to specify the hex equivalent such that "x" equals H,00 (ON) or H,01 (OFF).
$OTP=H,xx	; This addresses all 8 outputs simultaneously and must utilize hexadecimal format only. "xx" ranges from 00 (all ON) to FF (all OFF).

RETURNS:

EXAMPLE:

$OT5=0	; Output 5 is active ON.
$OT6=H,01	; Output 6 is inactive OFF.
$OTP=H,6D	; This is equivalent to binary representation 01101101 which means that Output 0=1 (OFF) etc.
$OTP=VAR1	; The output state is equal to the data value of VAR1.

$OTn

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #1 and 2 located in Appendix 3 of this manual.

NOTES:

1) On a standard I/O mounting rack assembly PB8, 16, or 24 **$OT0** resides at position 0.

2) These outputs are not latched, and should be considered write only. Once enabled, the opto rack assembly will remain in the set state until <reset>, power down, or readdressing occurs.

3) Additional I/O is available via the MST bus and PAMUX I/O channel options. (See the *Unidex 21's Options Manual* for further detail.)

4) An output may also be activated by utilizing output on axis trap parameter which may be set under Unidex 21's Main System Parameter #56. (See the *Unidex 21's User's Manual* for a detailed description of the Parameter Mode.) This output will also be negated for program faults.

RELATED COMMANDS:

$INn, DVAR, SIOC

$POT

NAME:

$POT - RS-232 Port Receive Buffer

FUNCTION:

The **$POT** command is a system variable that reads one element of data from the background RS-232 Port Buffer (refer to the **PORT** and **MALC** commands). Data elements are four bytes in length, with the top three bytes set to zero and the least significant byte containing the data byte received from the RS-232 port (in hexadecimal format).

FORMAT:

$POT<0>	; Get number of data elements which have been stored.
$POT<N>	; Get Nth data element which has been stored. N may range from 1 to the number returned by **$POT<0>**.

RETURNS:

EXAMPLE:

Assume the Port Buffer is active and contains the following data:
00000031 00000032 00000033 00000034, remainder not used

```
VAR1=$POT<0>    ; VAR1 = 4, the number of data elements stored
VAR1=$POT<1>    ; VAR1 = H, 00000031
VAR1=$POT<2>    ; VAR1 = H, 00000032
VAR1=$POT<3>    ; VAR1 = H, 00000033
VAR1=$POT<4>    ; VAR1 = H, 00000034
```

NOTES:

For a more complete example of the use of **$POT** and related commands, see the example under the **PORT** command.

RELATED COMMANDS:

MALC, PORT

$R

NAME:

$R - Read Command

FUNCTION:

The **$R** command is a system variable that reads one element of data from a Read file, then increments the pointer to the next available data. If retrace is being used, the **$R** command will move back one data element and read. Data elements are separated by a space or a comma.

FORMAT:

$R

$R may be used as any other variable.

RETURNS:

EXAMPLE:

Assume data file is open and contains the following:
0, 16, 22, 64.75

```
VAR1=$R          ; VAR1 = 0
VAR2=$R          ; VAR2 = 16
VAR1=$R          ; VAR1 = 22
VAR1=$R          ; VAR1 = 64.75
(MEND, R)        ; Reset the read pointer
VAR2=$R          ; VAR2 = 0
```

NOTES:

For a more complete example utilizing the **$R** and related commands, see the example under the **OPEN** command.

RELATED COMMANDS:

END, MEND, OPEN, WRIT

$RTP

NAME:

$RTP - Real Time Position Buffer

FUNCTION:

The **$RTP** command is a system variable that reads one element of data from the Real Time Position Buffer (refer to the **MALC** and **RETP** commands). Data elements are hexadecimal format. The values represent axis position in machine steps.

FORMAT:

$RTP<0>	; Get number of data elements which have been stored. This number is the product of the number of times data has been stored times the number of axes for which position information is being recorded.
$RTP<N>	; Get Nth data element which has been stored. N may range from 1 to the number returned by **$RTP**<0>.

RETURNS:

EXAMPLE:

Assume Real Time Position Buffer is active, enabled to fetch position for the X and Y axes, and contains the following data:

00000000 00000163 80000224 000FA175, remainder not used

```
VAR1=$RTP<0>              ; VAR1 = 4, the number of data elements stored
VAR1=BTF($RTP<1>)         ; VAR1 = 0.0, first X axis position recorded

VAR2=$RTP <2>             ; VAR2 = H, 00000163, first Y axis position recorded
VAR2=BTF(VAR2)            ; VAR2 = 355.0

VAR1=$RTP<3>              ; VAR1 = H, 80000224, second X axis position recorded
VAR1=BTF(VAR1)            ; VAR1 = 2147483100.0

VAR2=BTF($RTP<4>)         ; VAR2 = 1024373.0
```

$RTP

NOTES:

For a more complete example utilizing the **$RTP** and related commands, see the example under the **RETP** command.

RELATED COMMANDS:

MALC, RETP

$TOD

NAME:

$TOD - Time of Day System Variable

FUNCTION:

This command outputs the current time/date for the **COMM**, **CPAG**, **MSG**, and **WRIT** commands. Individual components may also be accessed as floating point numbers.

These may be used for data logging or Statistical Process Control (SPC).

FORMAT:

```
$TOD                    ; Full time/date information for the COMM, CPAG, MSG, and
                          WRIT commands only.

        or

$TOD<option>            ; Individual component of time/date information
  $TOD<Y>               ; Number of current year
  $TOD<M>               ; Number of current month
  $TOD<D>               ; Number of current day
  $TOD<H>               ; Number of current hour
  $TOD<m>               ; Number of current minute
  $TOD<S>               ; Number of current second
```

RETURNS:

EXAMPLES:

```
(MSG, #$TOD)            ; The message display shows: 24-SEP-2004 05:35:22

(MSG, #$TOD<Y>)         ; The message display shows: 2004

(MSG, #$TOD<M>)         ; The message display shows: 09

VAR1=#$TOD<H>+10        ; Variable VAR1 equals 15 if the hour is 05.
```

$TOD

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #2 located in Appendix 3 of this manual.

NOTES:

RELATED COMMANDS:

COMM, CPAG, MSG, WRIT

$000-$FAF

NAME:

$000-$FAF - Extended I/O Cability

FUNCTION:

Unidex 21 provides for extended Input/Output capabilities beyond the standard 16 inputs, 8 outputs, MST bus, and 3 User interrupts.

The extended I/O is accessed using the system I/O variable. This permits the User to utilize the Motorola I/O channel, integral PLC option, and Opto 22 PAMUX channel.

$000-$7FF is used to address the Motorola I/O channel.

$800-$F6F is used to address the PLC Dual-Port Memory locations.

$F70-$FAF is used to address the PAMUX I/O channels for reading or writing data.

FORMAT:

$000-$FAF	; Address entire byte of I/O
or	
$000**<0-7> - **$FAF<0-7>	; Address individual bit of I/O

RETURNS:

EXAMPLES:

VAR1=**$000**	; Lower byte of VAR1 is set to the status of the first eight I/O lines of the Motorola I/O channel at **$000**, upper 3 bytes of VAR1 are set to 0.
$VAR1=**$000**<3>	; Variable VAR1 is set to 1 if the status of the fourth I/O lines (line 3 of 0-7) of the Motorola I/O channel at address **$000** is active, or to 0 if it is not.
$700<7>=0	; Set I/O line 7 at **$700** to inactivate.

$000-$FAF

EXAMPLES (CON'T):

```
$701=H,72          ; Set I/O lines 0-7 at $701 to the following pattern: (01110010)
                   ;   Line 0 OFF
                   ;   Line 1 ON
                   ;   Line 2 OFF
                   ;   Line 3 OFF
                   ;   Line 4 ON
                   ;   Line 5 ON
                   ;   Line 6 ON
                   ;   Line 7 OFF
```

NOTES:

1) Refer to the *Unidex 21 Hardware Manual* and individual *Options Manual* for a complete description of each I/O group.

2) The Unidex 21's Math Package (see Chapter 4) may be utilized on I/O system variables.

RELATED COMMANDS:

DVAR, SIOC

ABTS

NAME:

ABTS - Abort Subroutine

FUNCTION:

Upon completion of a Subroutine, the Unidex 21 normally returns to the next block of the "Caller" Program (the program that called the Subroutine). The **ABTS** command allows the Subroutine to be aborted and program execution to continue from an entry point defined by the User.

FORMAT:

(**ABTS**,entry point)

RETURNS:

EXAMPLE:

```
(DFS,TEST
 .
 .
 (JUMP,ENT1,VAR1.EQ.2)      ; Jump to Entry Point ENT1 if variable
 (ABTS,ENT2)                ; VAR1 equals 2. If it does not, then
 (DENT,ENT1)                ; ABTS command will be executed.
 .
)
(DENT,ENT2)
```

This example controls a jump to the Entry Point ENT2 before completion of the Subroutine rather than returning to the Main Program.

ABTS

NOTES:

1) The **ABTS** may be used in one of four ways; directly, indirectly, conditionally direct, or conditionally indirect. An example of each follows:

(**ABTS**,ENT1)	; Aborts current Subroutine, jumps to Entry Point ENT1.
(**ABTS**,#VAR1)	; Variable (VAR1) contains the ASCII name of the entry point.
(**ABTS**,ENT1, VARA.EQ.1)	; Program jumps to Entry Point ENT1 if VARA equals 1.
(**ABTS**,#VAR1,VARA.EQ.1)	; Program jumps to entry point defined by VAR1 if VARA equals 1.

2) **ABTS** is not a modal command.

3) The former return address is removed from the program stack.

RELATED COMMANDS:

CLS, DENT, DFLS, DFS, INT1/INT2

ACDE

NAME:

ACDE - Maximum Acceleration/Deceleration

FUNCTION:

The **ACDE** command provides the User with the ability to set the Acceleration/Deceleration rate for each axis, overriding the Parameter Mode setting of the maximum Acceleration and Deceleration rates. The Acceleration/Deceleration Parameter Mode settings are not changed by this command. The rates set with this command remain in effect until updated by a subsequent **ACDE** command, or the system is reset.

The **ACDE** rate effects only the jog functions (keyboard arrow keys) in both velocity and step modes, **FREE** (cases 2+3), **FXOF**, **G0**, **HOME** (REF), and **MORG** commands.

FORMAT:

(**ACDE**, axis name and value, axis name and value, ...)

RETURNS:

EXAMPLES:

```
(ACDE,X1000000,Y=VAR1,Z=ARY<3>)     ; This sets the Acceleration/Deceleration rate
                                    ; of the X, Y, and Z axes utilizing numeric
                                    ; value, variables value, and an array element
                                    ; respectively.

(ACDE,D)                            ; This returns the Acceleration/Deceleration
                                    ;  rate of all axes to the values established in
                                    ;  the Parameter Mode.
```

ACDE

NOTES:

All Acceleration/Deceleration rates established by the **ACDE** command are set in machine steps/sec^2, no decimals, for each axis. This means the User must take into account the machines step resolution, before parameter scaling.

RELATED COMMANDS:

DVAR, FILT, FREE, FXOF, G0, HOME, INT1/INT2, MORG, RAMP, REF

AFCO

NAME:

AFCO - Auto-Focus

FUNCTION:

The **AFCO** command provides the User with Auto-Focus control, i.e., each axis has the ability to lock into any one of two inputs on the Unidex 21's Indexer board. The input may be quadrature, CW/CCW clock, or CL/DIR signals. The number of counts present in the Auto-Focus register determines the velocity which the axes will move. This is different from the **HWEL** command where each count causes a particular distance to be moved. The Auto-Focus conversion factor as established in the Individual Axis Parameter Mode may be programmed to a different factor with this command. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.) The parameter values are not changed by this command. The Auto-Focus mode set by this command remains in effect until a subsequent **AFCO** command is used, or the system is reset.

FORMAT:

(**AFCO**, axis name and input number, axis name and input number, ...)

or

(**AFCO**, axis name and input number, <conversion factor 1 to 255>)

RETURNS:

EXAMPLES:

(AFCO,X1,Y2)	; The X axis is locked to Input 1. ; The Y axis is locked to Input 2 such that a CW input signal will cause CCW motion. ; The current conversion factor will be used.
(AFCO,X1,Y1)	; Both the X and Y axes are locked to Input 1. ; The current conversion factor will be used.
(AFCO,X=VAR1)	; The X axis is locked to data of variable VAR1, where variable VAR1= 0,1,2. ; The current conversion factor will be used.

AFCO

EXAMPLES (CON'T):

```
(AFCO,X1,<3.5>)          ; The X axis is locked to Input 1.
                         ; The conversion factor is set to 3.5.

(AFCO,X0)                ; The X axis is unlocked from any input.
```

NOTES:

1) The **AFCO** command can reverse the clock input direction if the input number is preceded by a "-".

2) A single input may be locked to multiple axes.

3) The applicable Main System Parameters are #28 and #29. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

4) The applicable Individual Axis Parameters are #40 and #41 as established under the Individual Axis Parameters 401-408 grouping. (See also the *Unidex 21 User's Manual* for further detail regarding these Parameters.)

5) The conversion factor scales the **AFCO** input, where the parameter values may be multiplied by up to 255.

RELATED COMMANDS:

HWEL, SLEW

CCP

NAME:

CCP - Cutter Compensation

FUNCTION:

The Cutter Compensation command provides two axis setting and tool diameter information for **ICRC**. This information is used when **ICRC** is activated. The tool diameter value of **CCP** should be positive only. For more complete examples of the use of **CCP** with **G40**, **G41**, and **G42**, refer to the Intersectional Cutter Radius Compensation (**ICRC**) Appendix 1.

FORMAT:

(**CCP**,1st,2nd,nnn) ; 1st is the name of the first axis, 2nd is the name of the second axis, nnn is the tool diameter (programmed as a decimal number). The tool diameter may also be expressed as a variable.

A special format may be used in order to match the format recognized by the Unidex 16. The format is:

(**CCP**,T1=VAR) is the same as (**CCP**,X,Y,VAR)

(**CCP**,T1,nn) is the same as (**CCP**,X,Y,nn)

RETURNS:

EXAMPLES:

```
G70                    ; Initiates English Programming (inches).

(CCP,X,Y,1.5)          ; ICRC is on the X/Y plane with a tool diameter of 1.5 inches.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #5 located in Appendix 3 of this manual.

CCP

NOTES:

1) **ICRC** is valid for Linear and Circular Interpolation only.

2) The **CCP** command is a modal command, i.e., any subsequent **CCP** command for a particular tool updates the tools **CCP** to the new value.

RELATED COMMANDS:

DVAR, G40, G41, G42

CLS

NAME:

CLS - Call Subroutine

FUNCTION:

The **CLS** command calls a User Defined Subroutine or a User Defined Library Subroutine.

FORMAT:

(CLS, subroutine name)

The Subroutine name may range from 2 to 4 characters in length. It should be defined uniquely by a **DFS** Define Subroutine or **DFLS** Define Library Subroutine command. The first two characters are letters from A-Z, the remaining two are alphanumeric characters.

A Subroutine may be called directly, such as:

```
(CLS,AB12)          ; Subroutine AB12 is called.
```

or indirectly:

```
(DVAR,VAR1)         ; A variable name defined as ASCII string, must be contained
                      within quotes.

(CLS,#VAR1)         ; The variable VAR1 contains the ASCII name of the Subrou-
                      tine.  In this example, it would call the Subroutine named
                      ABCD.
```

RETURNS:

The return address is kept in a User's stack. If the Subroutine is defined as a Library Subroutine, the modal commands are also kept in the stack.

EXAMPLE:

```
VAR1="SUB1"
(CLS,#VAR1)
M2
```

CLS

EXAMPLE (CON'T):

```
(DFS,SUB1
  .
  .
)

        This is the same as;

(CLS,SUB1)
M2
(DFS,SUB1
  .
  .
)
```

In the first example, #VAR1 tells the Unidex 21 that the Subroutine is defined by a variable. This method of defining subroutines allows flexibility in programming since the Subroutine being called can vary with the value of VAR1.

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #4 located in Appendix 3 of this manual.

NOTES:

1) All User defined variables are global. The Unidex 21 does not provide local variables; however, if a User Defined Variable is to be used as a local variable in a Subroutine, the value of the variable may be stored on the User stack with the **PUSH** instruction, and later restored with the **POP** instruction.

2) Calling a normal Subroutine requires 8 bytes of User memory. Calling a Library Subroutine requires 52 bytes of User memory.

RELATED COMMANDS:

ABTS, DFLS, DFS, POP, PUSH, STKP

COEF

NAME:

COEF - Coefficient for Parabolic Trajectory Ramping

FUNCTION:

The **COEF** command is used to select a Parabolic Trajectory Coefficient for Ramping. The coefficient may range from 0 to 65,535. The value established by this command will override the coefficient established in the Parameter Mode, but does not change the Parameter Mode setting. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.) The value of the Parabolic Trajectory Coefficient set with this command will remain in effect until a subsequent coefficient command is issued, or the system is reset.

FORMAT:

(**COEF**,axis and coefficient, axis and coefficient,...)

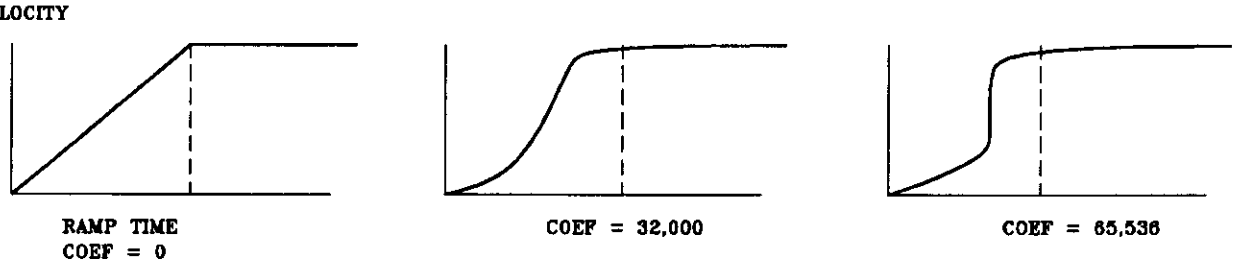
RETURNS:

EXAMPLE:

```
(COEF,X20., Y1000., Z=VAR1)
```

NOTES:

1) The value of the coefficient affects the shape of the Acceleration/Deceleration velocity profile. A coefficient of 0 becomes a linear ramp. Higher values result in a steeper curve as illustrated below.



2) The relevant Main System Parameters are #30, 57, and 58. The applicable Individual Axis Parameters are #9 and #38 as established under the 401-408 grouping.

RELATED COMMANDS:

ACDE, FILT, RAMP, TRAJ

COMM

NAME:

COMM - Communication through RS-232 Port A/B

FUNCTION:

The **COMM** command provides routing for data input or output through the RS-232 port within a program. This is particularly useful for controlling remote devices, and passing messages between external instruments such as Lasers and PLC's (Programmable Logic Controllers).

FORMAT:

(**COMM**,port,<optional input>, text)

The Unidex 21 recognizes the following format for output through RS-232:

#VAR	- as the decimal format of the variable
#H:VAR	- as the hexadecimal format of a variable
#C:VAR	- as the character format
#C:VAR1,VAR2,VAR3	- as a variable string

RETURNS:

Comm, 1,

EXAMPLES:

```
(COMM,A,...text)                    ; Data is sent through the RS-232, Port A.

(COMM,B,<VAR1,ARY<2>>, ...text..)   ; Data is sent and input data returned
                                      through the RS-232, Port B.

(COMM,A,#C:H,0D0A)                  ; Transmit the carriage return and line feed
                                      through the RS-232, Port A.
```

Input data may be floating numbers (123.4), hex (H,1F30), or characters (ABCD).

If an input string consists of more than four characters, "12345678", the string will be divided into groups of four characters; VAR1="1234" ARY<2>="5678". See the **MSG** command for more examples of output text formatting and use of input variables.

COMM

PROGRAMING EXAMPLE:

```
(DVAR,VAR1,VAR2)                    ; Define variables VAR1 and VAR2.
(COMM,A,Aerotech)                   ; This sends "Aerotech" through
                                    ; the RS-232, Port A.
(COMM,B,Aerotech)                   ; This sends "Aerotech" through
                                    ; the RS-232, Port B.
(COMM,B,#C:H,0D0A)                  ; This transmits carriage returns
                                    ; and line feed on RS-232, Port B.
(COMM,A,<VAR1>, "NEXT")             ; Character data is sent and input data
                                    ; returned on the RS-232, Port A.
(MSG,#C:VAR1)                       ; Print the character format variable
                                    ; VAR1 to the screen.
(COMM,A,<VAR2>, input 4 numbers)    ; Data is sent and data returned
                                    ; on the RS-232, Port A.
(MSG,#VAR2)                         ; Print the decimal format variable
                                    ; to the screen.
M0                                  ; Program Stop
```

NOTES:

1) Prior to inputting data, adequate variable names must be used to accommodate an input string.

2) Echo status for RS-232 input data is established within the Parameter Mode. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

3) Variables may contain 4 alphabetic characters maximum and may be strung together to larger input strings.

4) The Unidex 21 will wait until data is received before proceeding with the next program block. Data validity is not checked. However, if necessary, this may be done using a separate command(s).

RELATED COMMANDS:

DVAR, MSG

CPAG

NAME:

CPAG - Customer Display Page

FUNCTION:

The **CPAG** command provides the User the ability to replace the Machine/Debug display with a Customer Designed Display Page. This can be useful to provide menu driven operation with highly visible and customized prompts. When a page is active you can toggle it on/off by typing <alt H> to return to the normal machine mode display. Note that memory must be allocated via the **MALC** command before trying to create **CPAG**.

FORMAT:

(**CPAG**,option 1,option 2,....)

Options may be either a variable or an array.

(**CPAG**,0) ; Deactivates the Customer Display Page and returns to the Machine/Debug Display.

(**CPAG**,1) ; Activates the Customer Display Page.

(**CPAG**,2,x1,y1,x2,y2,color) ; Draws a box of the specified color with <x1,y1> as the upper left corner and <x2,y2> as the lower right corner.

x : 0 to 79 (80 columns)
y: 0 to 23 (24 rows)

color code	background	text
0	black	white
1	blue	yellow
2	green	magenta
3	cyan	red
4	red	cyan
5	magenta	green
6	yellow	blue
7	white	black

CPAG

FORMAT (CON'T):

(**CPAG**,3,x1,y1,x2,y2,char) ; This draws a box filled with the specified character with <x1, y1> as the upper left corner, and <x2, y2> as the lower right corner. The character is fixed (i.e., cannot be a variable or array element).

(**CPAG**,4,x1,y1,string) ; Starts from x1, y1. Display string is in 1x1 format. The string is fixed and can't be a variable or array element.

(**CPAG**,5,x1,y1,string) ; Starts from x1, y1. Display string is in 2x2 format. The string is fixed and can't be a variable or array element.

(**CPAG**,6,x1,y1,Variable/Array) ; Starts from x1,y1 to display the contents of variable or array element in 1x1 format.

(**CPAG**,7,x1,y1,Variable/Array) ; Starts from x1,y1 to display the contents of variable or array element in 2x2 format.

(**CPAG**,8,set,case,format,size,adjust,x1,y1,system variable) ; Enables a 200 msec periodic update of information.

; Set: 1 to mmm (see MALC)
; Case: 1 1x1 character size
; 2 2x2 character size

0 disables this set of functions, no code follows, so (**CPAG**,8,set,0) is a complete code

Format: 1 – unsigned Hex number

0 – unsigned Binary number

1 – BCD, 1 digit after decimal point

2 – BCD, 2 digits after decimal point

3 – BCD, 3 digits after decimal point

N – BCD, N digits after decimal point

CPAG

FORMAT (CON'T):

```
                                    ; Size: space needed for character display (<127)
                                    ; Adjust: 0 starts from left, with leading zero
                                    ; <> 0 start from right no leading zero
                                    ; x1, y1: start from this point
                                    ; System variable: axis position or 1/0
(CPAG,9,set,case,y1,nn,mm)          ; For 200 msec display of User's Parts Program
                                    ; Set: 1 to mmm (see MALC)
                                    ; Case: 0-disable, <> 0 enable
                                    ; y1: current block display
                                    ; Line # on which to display current block
                                    ; nn: number of blocks above current block
                                    ; mm: number of blocks below current block
```

0 disables this set of functions, so (**CPAG**,9,set,0) is a complete code.

RETURNS:

EXAMPLE:

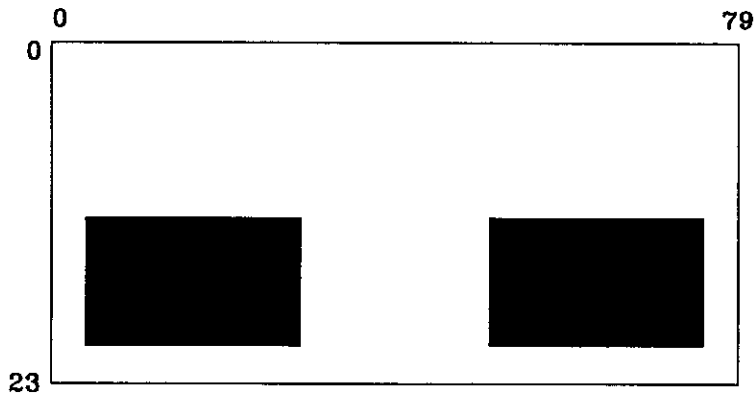
PROGRAMMING EXAMPLE #1:

The following is an example illustrating the functionality of the **CPAG** command showing the effect on a shadow of the screen. The actual image will not be displayed until the Customer Defined Page is activated.

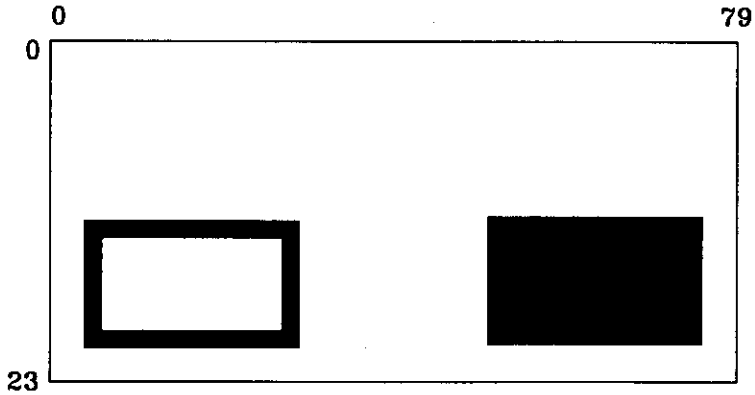
```
(DVAR,VAR1)                 ; The variable VAR1 holds the interrupt count.
(CPAG,2,4,12,26,20,7)       ; A box will be drawn in reverse video with corners at
                            ; 4,12, and 24,20.
(CPAG,2,55,12,75,20,7)      ; An additional box will be drawn with corners at 55,12
                            ; and 75,20.
                            ; Refer to the figure on the following page.
```

CPAG

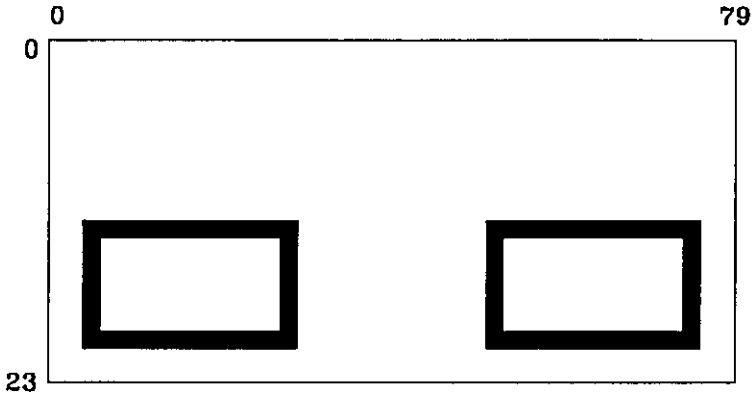
PROGRAMMING EXAMPLE #1 (CON'T):



(**CPAG**,2,6,14,24,18,0) ; A normal colored box is drawn within the 1st box,
; (see the figure below)



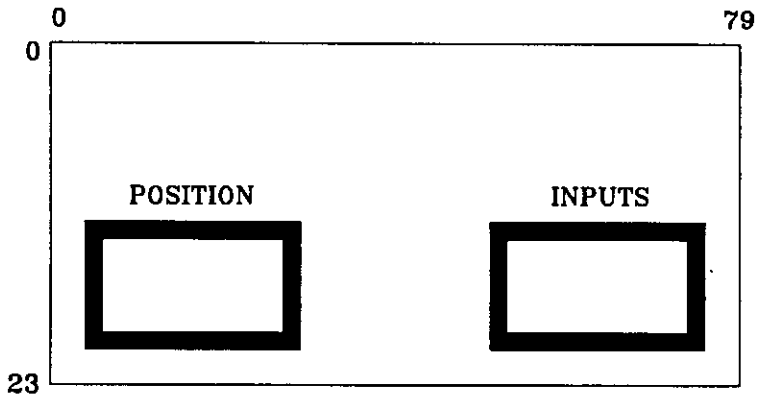
(**CPAG**,2,59,15,71,17,0) ; and another within the 2nd box.
; (See the Figure below.)



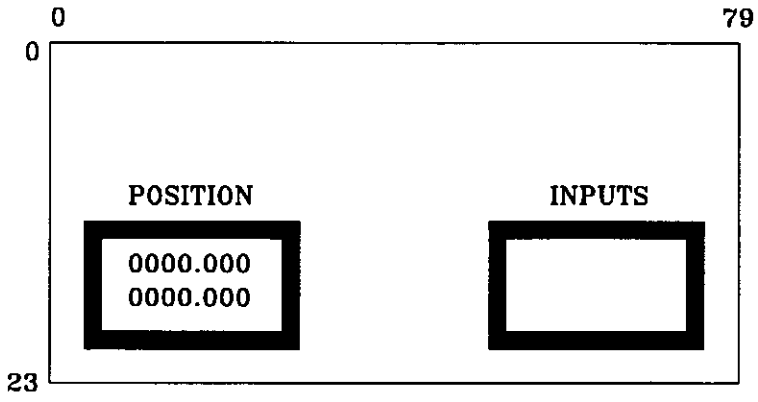
CPAG

PROGRAMMING EXAMPLE #1 (CON'T):

```
(CPAG,5,8,10,POSITION)      ; Display the word position (2x2 format) above
                            ; the 1st box, and I/O above the second box
(CPAG,5,60,10,INPUTS)       ; (same format).
                            ; See the Figure below.
```



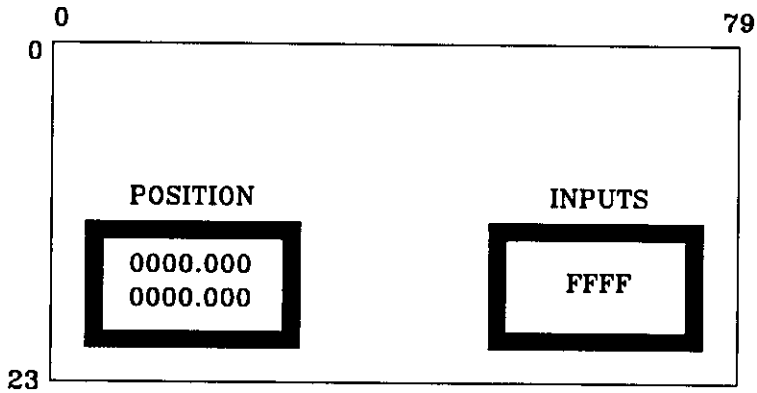

```
G92 X0. Y0.                 ; Reset all positions to 000.000.
(MALC,<1,4>)                ; Allocate memory for 4 CPAG items.
(CPAG,8,1,2,3,14,0,6,15,$XRP)  ; Display X axis Relative Position at 6,15. Use 2x2
                            ; format, BCD with 3 digits after decimal point, and
                            ; left justified (leading 0's).
(CPAG,8,2,3,14,0,6,17,$YRP)  ; Display Y axis Relative Position at 6,17. Format is
                            ; the same. Both are updated every 200 msec.
                            ; See the Figure below.
```



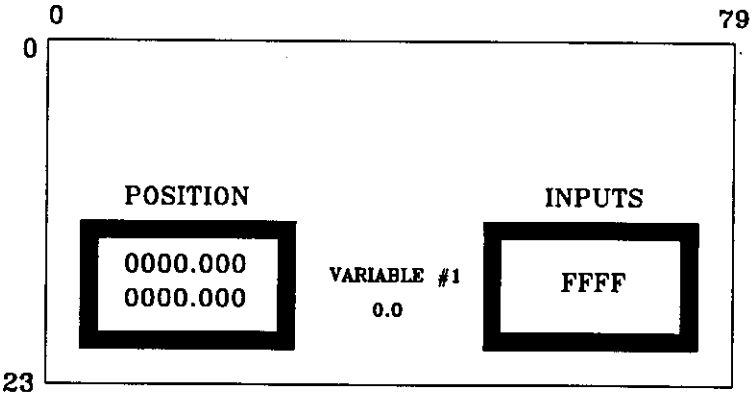
CPAG

PROGRAMMING EXAMPLE #1 (CON'T):

```
(CPAG,8,3,2,-1,4,1,62,16,$INP)   ; Display current inputs in hexadecimal at location
                                 ; 62,16. Uses 2x2 format. Right justified, maximum
                                 ; of 4 digits. Also updated every 200 msec.
                                 ; See the Figure below.
```




```
(CPAG,4,35,15,VAR1)              ; Display VAR1 at location 35,15 and the current
(CPAG,6,37,17,VAR1)              ; of VAR1 in 1x1 format at 37,17.
                                 ; (BCD with N digits after decimal)
                                 ; See the Figure below.
```

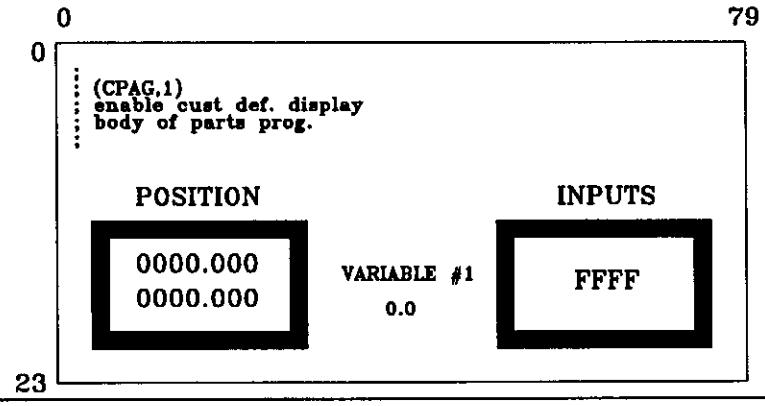



```
(CPAG,9,4,1,5,2,3)               ; Display the Parts Program with current line on line 5. Two
                                 ; lines will be displayed above, and three lines below.
                                 ; See the Figure below.
```

CPAG

PROGRAMMING EXAMPLE #1 (CON'T):

```
(CPAG,1)          ; Enable the Customer Defined Display body of the Parts
                  ; Program. X and Y Relative Position will be
                  ; continuously updated on display as will the inputs and
                  ; outputs. The value of variable VAR1 will be updated
                  ; only if the program issues another CPAG command.
(CPAG,0)          ; Disable the Customer Defined Display. Display normal
                  ; Machine Mode screen again.
M2                ; End of The Program.
```



PROGRAMMING EXAMPLE #2:

This example shows the various methods of using the **CPAG** command to display text, system variables, User variables, and the currently executing Parts Program. It also demonstrates how the program may receive input from the User while using the Customer Defined Page screen.

```
(MALC,<1,9>)                    ;
(CPAG,2,0,0,79,27,7)            ;
(CPAG,2,0,0,79,8,1)             ;
(CPAG,3,0,0,79,27, )            ;
(CPAG,5,1,1,HORZ)               ;
(CPAG,8,1,2,4,10,1,9,1,$XRP)    ;
(CPAG,5,1,3,VERT)               ;
(CPAG,8,2,2,4,10,1,9,3,$YRP)    ;
```

CPAG

PROGRAMMING EXAMPLE #2 (CON'T):

```
(CPAG,5,1,5,ROTY)                                    ;
(CPAG,8,3,2,4,10,1,9,5,$ZRP)                         ;
(CPAG,5,1,7,PROB)                                    ;
(CPAG,8,4,2,4,10,1,9,7,$URP)                         ;
(CPAG,5,41,1,AZIM)                                   ;
(CPAG,8,5,2,4,10,1,49,1,$xRP)                        ;
(CPAG,5,41,3,PLC1)                                   ;
(CPAG,8,6,2,4,10,1,49,3,$INP)                        ;
(CPAG,5,41,5,PLC2)                                   ;
(CPAG,8,7,2,4,10,1,49,5,$INP)                        ;
(CPAG,5,41,7,PLC3)                                   ;
(CPAG,8,8,2,4,10,1,49,7,$ZAP)                        ;
(CPAG,2,0,11,79,22,4)                                ;
(CPAG,2,0,16,79,16,2)                                ;
(CPAG,9,9,1,16,5,6)                                  ;
(CPAG,5,3,24,LASER POWER=2.35Kw)                     ;
(CPAG,5,3,26,LASER GAS :ON ;SHUTTER :OPEN)           ;
(CPAG,1)                                             ;
M0                                                   ;
```

HORIZ	.5298	AZIM	.0787
VERT	.5271	PLC1	.9769
ROTY	.3314	PLC2	.9042
PROB	.6254	PLC3	.3792

```
(CPAG,2,0,16,79,16,2)                                ;
(CPAG,9,9,1,16,5,6)                                  ;
(CPAG,5,3,24,LASER POWER=2.35Kw)                     ;
(CPAG,5,3,26,LASER GAS :ON :SHUTTER :OPEN)           ;
(CPAG,1)                                             ;
M0                                                   ;
```

CPAG

NOTES:

1) A 200 msec Customer Display requires a Memory Allocation **MALC** prior to initiation.

2) Each data item is to be updated every 200 msec and must have a unique "set" number.

3) The applicable Main System Parameter is #50 (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode) .

4) The set parameter in **CPAG** options 8 and 9 associates the chosen system variable with a memory set allocated with (MALC,1,n).

RELATED COMMANDS:

DARY, DVAR, MALC

DARY

NAME:

DARY - Define Array

FUNCTION:

The **DARY** command is used to Define an Array. The array may have been created to save information into a data file for ease of element retrieval. Also, a table of either on/off points or individual fine points may be created for usage by the PSO option card.

FORMAT:

(**DARY**,array name 1 and size,array name 2 and size,....)

RETURNS:

EXAMPLE:

```
(DARY,AR1<n>,AR2<m>,..)          ; Defines n+1 elements of array named AR1
                                   and m+1 elements of array named AR2.
                                   ("n" and "m" non zero, positive are fixed
                                   integers. The array consists of "0" to "n/m"
                                   elements.)

(DARY,AR1<10>)                   ; Defines 11 elements of array named "AR1".
                                   The 11 programmed elements of the array
                                   may be referenced as AR1<0> through
                                   AR1<10>,

(DARY,AR1<3>)
AR1<0>=1
AR1<1>=AR1<0>+1
AR1<2>=AR1<1>*2
AR1<3>=10
(MSG, elements 0-3 of AR1 are: #AR1<0>, #AR1<1>, #AR1<2>, #AR1<3>)
```

In this example, the message display on the Unidex 21 would present:

"elements 0-3 of AR1 are: 1,2,4,10"

DARY

NOTES:

1) Array size is enclosed in < > not ( ).

2) By allocating n + 1 elements for an array, this command allows the programmer to number the first element of the array as "0" (0-(n-1) or as "1" (1-n) according to preference.

3) All elements of the array are initialized to zero.

4) The minimum array size has two elements: (DARY,ARY<1>) and contains elements ARY<0> and ARY<1>.

RELATED COMMANDS:

$R, DVAR, END, MEND, OPEN, WRIT

DENT

NAME:

DENT - Define Entry Point

FUNCTION:

The **DENT** command is used to Define an Entry Point for a **JUMP** or **INT1/INT2** command.

FORMAT:

(**DENT**,entry point name)

The entry point name may be 2 to 4 characters in length. The first two characters must be A-Z, the second two may be any alphanumeric characters although they cannot conflict with system commands.

RETURNS:

EXAMPLE:

```
(DENT,ENT1)
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples # 1, 2, 3, and 7 located in Appendix 3 of this manual.

NOTES:

1) The name used to Define an Entry Point must be a unique identification that is not used anywhere else in the program in a definition, and it cannot be a User variable.

2) The **DENT** command should occupy its own block within a program.

RELATED COMMANDS:

ABTS, INT1/INT2, JUMP

DFLS

NAME:

DFLS - Define Library Subroutine

FUNCTION:

The **DFLS** command is used to define a Library Subroutine. Defining a Library Subroutine is done in the same manner as defining a Subroutine. A Library Subroutine however, restores previous modal machine states upon it's completion (see Notes).

A Library Subroutine may be called by a **CLS** or **INT1/INT2** command.

FORMAT:

(**DFLS**,subroutine name

.

.

.

)

The Library Subroutine name may be 2 to 4 characters in length. The first two characters of the Library Subroutine name must be letters, A-Z, the second two may be any alphanumeric characters.

RETURNS:

The program will continue with the next program block following a **CLS** command or **INT1/INT2** call.

EXAMPLE:

```
M2
(DFLS,SB12
 S300 M3
 Z-400. F100.
 M5
 G4 F0.3
 M4
 Z500.
)
```

DFLS

NOTES:

1) The **DFLS** command should occupy its own block within a program and be placed at the end of the Main Program following an **M2** command.

2) A program calling a Library Subroutine will not have its modal information altered after the Subroutine is executed. The modal commands which are reinstated after a Subroutine are:

   All modal **G** and **H** codes
   DRUN
   F
   MIR
   RTRS
   SCF
   SIOC
   UMFO

3) A **DFLS** command requires 8 bytes of User stack memory.

4) The name used to define a Subroutine must be a unique identification that is not used anywhere else in the program in a definition, and it cannot be a User variable. However, a **CLS** Call Subroutine command can be set up as a variable.

RELATED COMMANDS:

ABTS, CLS, DFS, INT1/INT2

DFS

NAME:

DFS - Define Subroutine

FUNCTION:

The **DFS** command is used to Define a Subroutine. Every Subroutine must be explicitly defined by a **DFS** command.

FORMAT:

(**DFS**,subroutine name

.

.

.

)

The Subroutine name may be 2 to 4 characters in length. The first two characters of the Subroutine name must be letters, A-Z, the second two must be alphanumeric characters.

RETURNS:

The program will continue with the next program block following a **CLS** command or **INT1/INT2** call.

EXAMPLE:

```
M2
(DFS,TAP1
 S500 M3
 Z-1000. F200.
 M5
 G4 F0.2
 M4
 Z1000.
)
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #4 located in Appendix 3 of this manual.

DFS

NOTES:

1) The **DFS** command should occupy its own block within a program and be placed at the end of the Main Program following an **M2** command.

2) The modal information of the program calling the Subroutine may be altered after execution of the Subroutine.

3) Subroutines may call other subroutines.

4) Each **DFS** command requires 8 bytes of User stack memory.

5) The name used to define a Subroutine must be a unique identification that is not used anywhere else in the program in a definition and it cannot be a User variable. However, a **CLS** Call Subroutine command can be set up as a variable.

RELATED COMMANDS:

ABTS, CLS, DFLS, INT1/INT2

DRUN

NAME:

DRUN - Dry Run

FUNCTION:

The **DRUN** command is used to enable or disable a Dry Run on an individual axis. Enabling an axis for Dry Run prevents that axis from performing program moves. This is useful for debugging a program i.e., disabling Z axis while permitting X and Y.

FORMAT:

(**DRUN**,axis name and enable/disable, axis name and enable/disable,...)

The Dry Run function is enabled by placing a 1 (non-zero) next to the axis name.

The Dry Run function is disabled by placing a 0 next to the axis name.

RETURNS:

EXAMPLE:

```
(DRUN,X0,Y1)                ; Disables a Dry Run on the X axis, and enables a Dry Run on
                              the Y axis.
```

Then, upon entering the program:

```
G2 G17 X10. Y10. I10. F100.  ; Only the X axis will move.

(DRUN,variable)              ; If the variable is 0, Dry Run is disabled; if the variable is
                              non-zero, Dry Run is enabled.
```

NOTES:

1) All other commands are processed as usual.

2) Similar results can be obtained by using the **MTOR** command.

RELATED COMMANDS:

MTOR

DVAR

NAME:

DVAR - Define Variable

FUNCTION:

The **DVAR** command is used to Define a Variable. The ability to utilize variables is probably the single most powerful programming feature. Almost every command and function may be set and changed through the use of variables. This enhances Parts Program versatility and saves program memory space. There are two types of variables, System Variables and Program Variables. System Variables are denoted by the **$nnn** system and are utilized for I/O, Position Registers, and general machine functions. The System Variables name may not be changed although the User can equate their value to another program variable. Program Variables are defined and utilized within a Parts Program, and must be defined prior to usage.

FORMAT:

(**DVAR**,variable name,variable name,...)

RETURNS:

EXAMPLE:

```
(DVAR,VAR1,VAR2,XYZ,BI34,...)
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #7 located in Appendix 3 of this manual.

NOTES:

1) The name used to Define a Variable must be a unique identification that is not used anywhere.

2) The variable name may range from 2 to 4 characters in length.

3) The first two characters of the variable name must be letters, A-Z, the second two may be any alphanumeric characters.

DVAR

NOTES (CON'T):

4) All variables defined occupy 4 bytes of User memory. These 4 bytes are initialized to zero, and can be used to hold either numeric or character data.

RELATED COMMANDS:

$ System Variables, CLS, COMM, CPAG, DARY, JUMP, MALC, TERM

DZON

NAME:

DZON - Define Safe Zone

FUNCTION:

The **DZON** command is used to Define the number of Safe Zones to be included within a program. Safe Zones define areas where axes can operate. An attempt to operate outside a Safe Zone will generate an error message and stop axis motion (see **ZONE** command).

FORMAT:

(DZON,n) ; The number of Safe Zones entered must be an integer, (not a variable or an array) ranging from 1 to 65,535, and should appear only one time at the beginning of the program. Internally, this number becomes n+1 (0 to n) sets of Safe Zones.

RETURNS:

EXAMPLE:

```
(DZON,15)        ; "16" sets of Safe Zones are defined in the program following
                   this command.
```

NOTES:

1) If the number of Safe Zones entered by this command is less than the actual number of Safe Zones contained in the program, the following error message will appear when the program begins to run:

   STATUS: error, undefined Safe Zone

2) The number of Safe Zones entered by this command must include any Safe Zones defined in programs "joined" with this one.

3) Safe Zone limits are not in effect when motion is initiated using the **HOME, FREE,** and **JOG/SLEW** commands. The User may want to utilize **LIMT** instead.

RELATED COMMANDS:

LIMT, ZONE

ELPS

NAME:

ELPS - Ellipse Look-Up Table

FUNCTION:

The Unidex 21 provides Elliptical Trajectory programming which is useful for laser hole cutting in a tubular structure or x-y flat sheet processing. The ellipse is executed at constant surface speed as opposed to other controllers which scale circles such that the resultant velocity varies widely. The **ELPS** command is used to generate a Look-Up Table to be used when performing an Elliptical Trajectory. At the present time, two Look-Up Tables may be stored in memory on the Unidex 21's Indexer board. The Look-Up Table is necessary to maintain constant surface speed while performing elliptical motion. The **ELPS** command may take several seconds for the initial completion of the Look-Up Table, after which execution as called by the **G2**, and **G3** commands occurs immediately.

FORMAT:

(**ELPS**,g,Xa,Yb)

g= the ellipse identification number. This number must be used to reference this ellipse.

X= represents the axis name from which conversion factors will be referenced

a= length of the horizontal axis in program units

Y= represents the axis name from which conversion factors will be referenced

b= length of the vertical axis in program units

RETURNS:

EXAMPLE:

```
(ELPS,1,X10.,Y20.)        ; A Look-Up Table will be generated for Ellipse #1 with a
                            horizontal axis length of 10 program units, and a vertical axis
                            length of 20 program units.
```

Refer to the **G2** command for a programming example of drawing an ellipse.

ELPS

PROGRAMMING EXAMPLE:

```
; ***************************************ELLIP
(REF,X,Y)                    ; Establishes an X and Y Home position.
G70                          ; Initiate English Programming (inches).
G91                          ; Initiate Incremental Positioning.
G1 X10. Y35. F200.           ; Linear Offset Move.
(ELPS,1,X10.,Y20.)           ; Generate an Ellipse Look-Up Table.
G2,1,45,90,90,X0. Y0. F100.  ; Initiate a G2 Ellipse command.
M0                           ; Program Stop
; ***************************************SMALL ELLIPSE
(REF,X,Y)                    ; Establishes an X and Y Home position.
G70                          ; Initiate English programming (inches)
G91                          ; Initiate Incremental Positioning.
G1 X10. Y35. F200.           ; Linear Offset Move.
(ELPS,1,X2.,Y4.)             ; Generate an Ellipse Look-Up Table.
G2,1,45,90,90,X0. Y0. F100.  ; Initiate a G2 Ellipse command.
M0                           ; Program Stop
```

NOTES:

1) The **ELPS** command needs to be executed only once for a given ellipse. If the major and/or minor axes lengths change, the command must be reentered.

2) The Look-Up Table established for a given ellipse is retained in memory on the Unidex 21 Indexer board , even through a power down, until changed.

3) Actual elliptical motion is executed via the **G2** and **G3** commands.

4) Minor block execution delays are possible when partial ellipses are executed or if parts rotation is utilized.

RELATED COMMANDS:

G2/G12/H2/H12, G3/G13/H3/H13

F

NAME:

F - Define Axis Feedrate

FUNCTION:

The **F** command is used to Define the Axis Feedrate in inches (**G70**) or millimeters (**G71**) per minute. Rotary axes will be specified in units per minute. The **F** command indicates the vectorial Feedrate of a move and applies to **G0, G1, G2, G3, G5,** and **FREE** axis commands.

FORMAT:

Feedrate values may be designated in two ways:

without a "." **F**1234	; The system always places a decimal point two digits from right. The Feedrate will be 12.34 inches or millimeters per minute.
with a "." **F**1234.	; The system interprets the value as written. The Feedrate will be 1,234 inches or millimeters per minute.

RETURNS:

EXAMPLE:

```
G71 X10. F1000          ; The X axis will move 10 millimeters at a Feedrate of 10.00
                          millimeters per minute.
G70 X10. F1000.         ; The X axis will move 10 inches at a Feedrate of 1,000 inches
                          per minute.
(DVAR,FEED)
FEED=250.0
F=FEED                  ; The Feedrate is equal to 250 program units per minute.
```

NOTES:

1) The **F** command is modal. User Manual Feedrate Override (**MFO**) adjusts this modal Feedrate. The only exceptions are **G0** moves, where **MFO** is limited to a maximum 100% override.

F

NOTES (CON'T):

2) The programmed Feedrate is in units per minute and is dependent upon the setting of parameters #0, 1, 2, and 3 established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

$MFO, DVAR, FREE, G70, G71, UMFO

FILT

NAME:

FILT - Digital Filter

FUNCTION:

The **FILT** command is used to enable or override digital filtering which can occur on the millisecond position/velocity trajectory command being sent to the Unidex 21's DSP Servo Control card. This command is modal, and remains until changed or the system is reset.

The Unidex 21 provides a high performance PIDF Proportional, Integral, Derivative Servo Loop that also implements two feedforward gains: velocity and acceleration. As a result, the system follows (tracks) very closely to the actual command with minimal servo following error or lag. When moves are programmed with tangency, i.e., corners are programmed with small arcs, the Unidex 21 will smoothly follow the path under **G8** Velocity Profiling.

However, there exists a large base of CNC's which utilize an error based type of servo loop closure. In this case, programmed moves must not necessarily have tangency and corners but can be programmed as two linear moves. When executed under **G23** Corner Rounding Mode, these systems will inherently round the corners and provide relatively continuous velocity contouring even with non-tangent moves. Please refer to **G8**, **G9**, **G23**, and **G24** program commands for further information.

By utilizing the **FILT** command and associated Main System Parameters #60 and #61, the Unidex 21 is able to digitally replicate more traditional CNC type control functionality. This is of particular benefit for Smart 1 and Unidex 16 users, who may more easily utilize their programs on the Unidex 21. Also, CAD/CAM systems with post processor converters are more readily compatible with the Unidex 21.

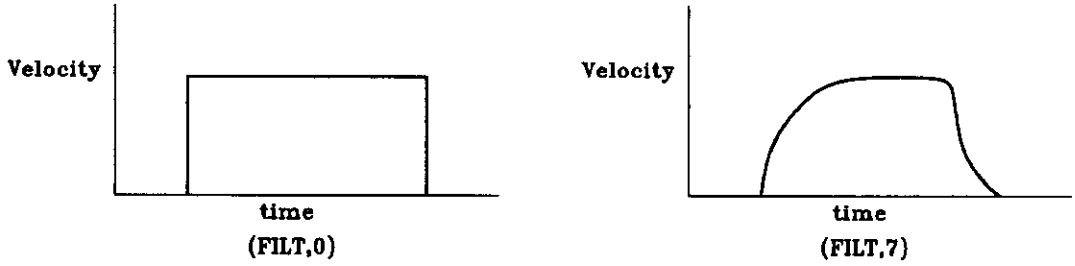
The digital filter works under the following relationships:

Filter Total = Filter Total + Command In
Command Out = Filter Total/2^n; where "n" equals FILT#
Filter Total = Filter Total - Command Out

This provides an exponential ramp to the motion, eliminating the need for **RAMP**. However, the User may select to program these as a supplement.

FILT

The values for **FILT** range from 0 (no filter) to 7 (maximum filter). The extreme spectrum would be:



point-to-point move assuming no Acceleration/Deceleration commands.

For normal systems a filter value of 4, 5, or 6 usually provides the best performance.

FORMAT:

(**FILT**,n) ; Where "n" ranges from 0-7

RETURNS:

EXAMPLES:

(FILT,0)	; Turn OFF the Digital Filter.
(FILT,VAR1)	; Set the Digital Filter to the value of VAR1. (range 0-7)

NOTES:

1) The Digital Filter cannot be turned OFF while the axes are moving or a sharp velocity transition may result. It should be turned OFF only after **G9** or **G24** type motion.

2) To disable axis ramping when operating in the **FILT** mode, axis parameter #61 must be set to "NO" (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode).

RELATED COMMANDS:

FREE

NAME:

FREE - Axis Free Run (enabled or disabled)

FUNCTION:

The **FREE** command is used to enable or disable Axis Free Run. This command is sent directly to the Unidex 21's DSP Servo Control card for execution. Therefore, no synchronization between axis contouring and the Free Run axis is possible. Once started, the Free Run command can cause motion to occur asynchronously from the User Parts Program flow. The Free Run command block execution time is 1ms and may be inserted between normal motion blocks to speed up program flow.

FORMAT:

(**FREE**,axis name and case, F ± Feedrate, D distance)

The code is selected as follows:

code 0 – stop Free Run

1 – one direction Free Run, with F + (CW), F - (CCW) Feedrate

2 – one shot Free Run, with F + (CW), F - (CCW) Feedrate and D distance

3+ n – back and forth Free Run, with F + (CW), F - (CCW) Feedrate and D distance
if n=0, Free Run back and forth an infinite number of times
if n=1 to 16777215, Free Run back and forth n times

(**FREE**,D,axis,MST data) ; When Axis Free Run is complete, output **M**,**S**,**T** data, at most four **M**,**S**,**T** commands.

(**FREE**,D,axis) ; Disable the **M**,**S**,**T** output function.

(**FREE**,W,axis1,axis2,...) ; Wait for completion of the axes Free Run before proceeding with the next program block.

RETURNS:

FREE

EXAMPLES:

(FREE,X1,F100.)	; Free Run is enabled for the X axis at a Feedrate of 100 inches or millimeters per minute in the CW (+) direction.
(FREE,X1,F-100.)	; Free Run is enabled for the X axis at a Feedrate of 100 inches or millimeters per minute in the CCW (-) direction.
(FREE,X0)	; Free Run is stopped for the X axis.
(FREE,Y=VAR1,F=VAR2)	; Free Run status and Feedrate are contained in variables VAR1 and VAR2.
(FREE,X2,F100.,D20.)	; A one shot Free Run is enabled for the X axis at a Feedrate of 100 inches or millimeters per minute in the CW (+) direction for a distance of 20 inches or millimeters.
(FREE,X3,F100.,D30.)	; A back and forth type Free Run is enabled for the X axis at a Feedrate of 100 inches or millimeters per minute starting in the CW (+) direction for a distance of 30 inches or millimeters.
(FREE,X3,F-100.,D30.)	; Same as above except moves CCW (-) direction first.
(FREE,X3+5,F-100.,D30.)	; Same as above except moves back and forth five times.
(FREE,D,X,M80,S2048,T101,M91)	; Upon completion of the X axis Free Run output M80, S2048, T101, and M91.

FREE

NOTES:

1) The **FREE** command is not modal. The CW (+) and CCW (-) direction are as specified in the axis setup of the Parameter Mode.

2) Case: 1 Free Run makes use of ramp time.

3) Case: 2 and 3 Free Run use **ACDE** rate or the Individual Axis Parameter #9 value as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

4) The Feedrate specified is only used during the Free Run and has no effect on the modal Feedrate parameter.

5) Manual Feedrate Override and Feedhold may be set up to also affect the Free Run axes. Refer to parameters #19 and 36 as established under the Individual Axis Parameters 401-408 grouping. (See also the *Unidex 21 User's Manual* for further detail on these parameters.)

RELATED COMMANDS:

ACDE, COEF, DVAR, F, RAMP, TRAJ

FXOF

NAME:

FXOF - Fixture Offset

FUNCTION:

The **FXOF** command provides the User the ability to move the axes to a new coordinate frame to accommodate a Fixture Offset. This position is an absolute distance from the Hardware Home (including any offsets) and automatically updates the **$nAP** Position Registers, while keeping the **$nRP** Position Registers at current values.

FORMAT:

(**FXOF**,axis name and new position,axis name and new position,....)

RETURNS:

$nAP Position Register value changes to reflect the new position as referenced to the Hardware Home Position; however, the **$nRP** Position Register value remains unchanged.

EXAMPLE:

```
(FXOF,X10.,Y=VAR1,Z=ARY<3>)          ; Moves the X, Y, and Z axes to a new coordi-
                                       nate frame located at (10.0,VAR1,ARY<3>).
```

NOTES:

1) All positions are referenced to the Hardware Home.

2) The axis speed is established in the "Top Feedrate steps/sec" Parameter #8 of the Individual Axis Parameters 401-408 grouping (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode).

3) The **FXOF** command executes point-to-point motion only, no contouring.

4) See the **$nAP/$nRP** command description and program example for further explanation.

RELATED COMMANDS:

$nAP, $nRP, G0, MORG, ROTA

G0

NAME:

G0 - Point-to-Point Positioning at rapid traverse rate

FUNCTION:

The **G0** command sets up axis movement for point-to-point positioning at a rapid traverse rate. The rate is determined by the "Top Feedrate counts/sec" Parameter #8 as established under the Individual Axis Parameters 401-408 grouping (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode). The Acceleration/Deceleration Ramping Trajectory for **G0** is always parabolic.

FORMAT:

G0 axis name and move distance

RETURNS:

EXAMPLE:

```
G0 X10.                    ; Moves to X10. using top system Feedrate.

G0 X5. Y10.                ;
```

NOTES:

1) The **G0** command is modal.

2) The **G8** command cannot be used in conjunction with the **G0** command.

3) The **G0** feedrate is limited to 100% MFO maximum.

4) The applicable Individual Axis Parameters are #8, 9, and 38 as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

COEF, FXOF, MORG

G1/G11/H1/H11

NAME:

G1/G11/H1/H11 - Linear Contouring

FUNCTION:

The **G1/G11/H1/H11** commands initiate Linear Contouring. As an example, if a move of X1. Y1. were programmed under **G0** motion, the axes would perform that move at their respective top feedrates uncoordinated. **G1** ensures that the axes will move synchronization at the programmed vectorial feedrate.

FORMAT:

G1 or **G11** or **H1** or **H11**

RETURNS:

EXAMPLE:

```
G17 G90 G70 G1 X4.0 Y3.0 F50.    ; A straight line will be produced from the initial
                                   position to the X=4.0, Y=3.0 coordinate po-
                                   sition, at a Feedrate of 50 inches per minute.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples # 1, 2, 5, and 6 located in Appendix 3 of this manual.

NOTES:

1) Prior to selecting a code for Linear Contouring, the appropriate axes/plane relationship must have been established (**G17/H17, G18/H18, G19/H19**).

2) The following code/plane relationships exist:

G1 - 1st plane axes	**H1** - 3rd plane axes
G11 - 2nd plane axes	**H11** - 4th plane axes

G1/G11/H1/H11

NOTES (CON'T):

3) The Contour Feedrate command (**F**) must be used in conjunction with the **G1/G11/H1/H11** commands.

4) The **G1/G11/H1/H11** commands are modal.

5) When transferring from lines to arcs and back again the User must reestablish **G1** mode of operation

   or

   a "circle missed center point" error will result.

RELATED COMMANDS:

F, FILT, G17/H17, G18/H18, G19/H19, G70, G71, G90, G91, RAMP

G2/G12/H2/H12

NAME:

G2/G12/H2/H12 - CW Circular Contouring

FUNCTION:

The **G2/G12/H2/H12** commands initiate Circular Contouring in the CW direction.

FORMAT:

G2 or **G12** or **H2** or **H12**

RETURNS:

EXAMPLE:

G17 G90 G70 **G2** X4.0 Y3.0 I2.0 J1.5 F50.	; A CW arc will be produced from the initial position to the X=4.0, Y=3.0 coordinate position, the center points being 2.0 and 1.5 (offset from starting point) at a Feedrate of 50 inches per minute.
G2 L10.0 C35.0 D70.0 F50.	; A CW arc with a 10 inch radius will be produced from the initial position of 35° and ending at 70°.

For more detailed examples, see **I,J,K,P/i,j,k,p** and **LCD,OAB/lcd,oab** commands.

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #1 located in Appendix 3 of this manual.

NOTES:

1) Prior to selecting a code for Circular Contouring, the appropriate axes/plane relationship must have been established (**G17/H17, G18/H18, G19/H19**).

2) The following code/plane relationships exist:

G2 - 1st plane axes	**H2** - 3rd plane axes
G12 - 2nd plane axes	**H12** - 4th plane axes

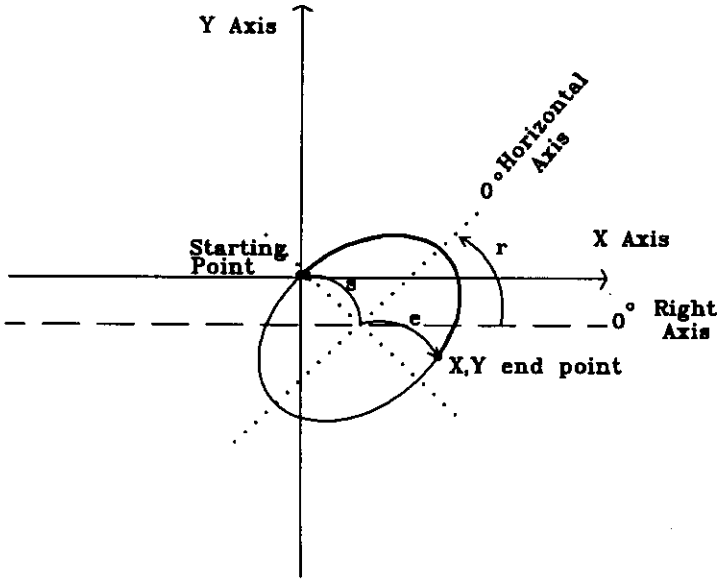
G2/G12/H2/H12

NOTES (CON'T):

3) Helical Interpolation is accomplished by programming two axes to do Circular Interpolation and a third axis to do Linear Interpolation.

4) Spherical Interpolation is accomplished by programming two axes to do Circular Interpolation and then programming the third and a hidden fourth axis to also do Circular Interpolation.

5) An Elliptical Contour is accomplished using the Circular Contouring command in conjunction with the following special code:
G,g,r,s,e,Zn Xa Yb Zd F



G– specifies axis plane and direction of rotation (**G2/G3/G12/G13/H2/H3/H12/H13**)

g – ellipse identification number (See **ELPS** command)

r – rotation angle of ellipse (in degrees) with respect to the right axis 0°

s – starting angle of ellipse (in degrees) with respect to the horizontal axis 0°

G2/G12/H2/H12

NOTES (CON'T):

e – ending angle of ellipse (in degrees) with respect to the horizontal axis 0°

Z – third axis name (omit if not used)

n – the options for n are as follows:

0 - third axis motion is not synchronized with axes performing the ellipse

1 - third axis moves with constant speed and is synchronized with axes performing the ellipse

2 - third axis moves proportionally to the angular velocity of the axes performing the ellipse

3 - third axes follows a tangent to the axes performing the ellipse

Xa Yb – axis name and calculated endpoint of the move (calculated by User). Note that if both end points (a,b) are zero, a full ellipse that begins and ends with the starting point will be generated.

d – distance of the third axis move

F – surface velocity of the ellipse

An example of Elliptical Contouring follows:

```
G2,1,45,90,90,Z3 X0. Y0. Z10. F100.     ; A CW full ellipse will be produced using
                                          Look-Up Table 1, with a 45° angle of rotation,
                                          starting and ending at 90° with a tangentially
                                          synchronized third axis.
```

6) It is not necessary for both axes involved in a Circular Contour to have the same resolution.

G2/G12/H2/H12

NOTES (CON'T):

7) The Contour Feedrate command (**F**) must be used in conjunction with the **G2/G12/H2/H12** commands.

8) The **G2/G12/H2/H12** commands are modal.

9) When transferring from lines to arcs and back again the User must reestablish **G1** mode of operation

or

a "circle missed center point" error will result.

RELATED COMMANDS:

ELPS, F, G3/G13/H3/H13, G17/H17, G18/H18, G19/H19
I/J/K/P and **i/j/k/p,**
L/C/D, O/A/B and **l/c/d, o/a/b,**
PLNE

G3/G13/H3/H13

NAME:

G3/G13/H3/H13 - CCW Circular Contouring

FUNCTION:

The **G3/G13/H3/H13** commands initiate Circular Contouring in the CCW direction.

FORMAT:

G3 or **G13** or **H3** or **H13**

RETURNS:

EXAMPLE:

G17 G90 G70 **G3** X4.0 Y3.0 I2.0 J1.5 F50.	; A CCW arc will be produced from the initial position to the X=4.0, Y=3.0 coordinate position, the center points being 2.0 and 1.5 (offset from starting point) at a Feedrate of 50 inches per minute.
G3 L12.0 C90.0 D180.0 F50.	; A CCW arc will be produced starting at 90°, ending at 180° at a Feedrate of 50 inches per minute.

For more detailed examples, see **I/J/K/P, i/j/k/p** and **LCD, OAB, lcd,oab** commands.

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #1 located in Appendix 3 of this manual.

NOTES:

1) Prior to selecting a code for Circular Contouring, the appropriate axes/plane relationship must have been established (**G17/H17, G18/H18, G19/H19**).

2) The following code/plane relationships exist:

G3 - 1st plane axes	**H3** - 3rd plane axes
G13 - 2nd plane axes	**H13** - 4th plane axes

G3/G13/H3/H13

NOTES (CON'T):

3) Helical Interpolation is accomplished by programming two axes to do Circular Interpolation and a third axis to do Linear Interpolation.

4) Spherical Interpolation is accomplished by programming two axes to do Circular Interpolation and then programming the third and a hidden fourth axis to also do Circular Interpolation.

5) An Elliptical Contour is accomplished using the Circular Contouring command in conjunction with a special code. (Refer to Note 5 of the **G2** command for a description and example of the Elliptical Contour code.)

6) It is not necessary for both axes involved in a Circular Contour to have the same resolution.

7) The Contour Feedrate command (**F**) must be used in conjunction with the **G3/G13/H3/H13** commands.

8) The **G3/G13/H3/H13** commands are modal.

9) When transferring from lines to arcs and back again the User must reestablish **G1** mode of operation

or

a "circle missed center point" error will result.

RELATED COMMANDS:

ELPS, F, G17/H17, G18/H18, G19/H19,
I/J/K/P and **i/j/k/p,**
L/C/D, O/A/B and **l/c/d, o/a/b,**
PLNE

G4

NAME:

G4 - Dwell

FUNCTION:

The Dwell command provides a time delay of a programmed or established duration. The **G4** command must be immediately followed by an **F** command to establish the Dwell duration. The resolution of the **G4** command is 0.1 seconds.

FORMAT:

G4 Fnn

The Dwell duration may be entered in either decimal or integer form. If the Dwell duration value is in decimal form (followed by a ".") or in the form of a variable, the Dwell will be in seconds. If the Dwell duration value is in integer form (not followed by a ".") the Dwell will be in tenths of seconds.

RETURNS:

EXAMPLE:

```
G4 F5.                  ; A Dwell of 5 seconds is established.

G4 F.5                  ; A Dwell of 5/10 (1/2) second is established.
      or
G4 F5.

G4 F=VAR1               ; The Dwell time is set to the value of variable VAR1.

G4 F=ARY<n>             ; The Dwell time is located in ARY<n>.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #2, 3, and 4 located in Appendix 3 of this manual.

G4

NOTES:

1) The Dwell command must occupy its own block within a program.

2) System interrupts or aborts are not active during dwells but will be executed immediately following the Dwell completion.

RELATED COMMANDS:

G5/G15/H5/H15

NAME:

G5/G15/H5/H15 - Three Point Interpolation of two dimensional arc for contouring

FUNCTION:

The **G5/G15/H5/H15** commands initiate Three Point Interpolation of a two dimensional arc for contouring. Utilizing these commands will alert the Unidex 21 that the values of **I/J/K/P** and **i/j/k/p** represent points on the arc, not center points. The arc direction is internally derived. The **G5/G15/H5/H15** command is utilized when digitized data is acquired from the Trackball/Joystick/Teachpendant options.

FORMAT:

G5 or **G15** or **H5** or **H15**

RETURNS:

System automatically resets to **G2/G3**, **H2/H3**, **G12/G13**, or **H12/H13** mode.

EXAMPLE:

G17 G91 G70 **G5** X4.0 Y0. I2.0 J2.0 F100.	; A CW arc will be produced from the initial position to the X=4.0, Y=0 coordinate position crossing points I=2.0, J=2.0 at a Feedrate of 100 inches per minute.

PROGRAMMING EXAMPLE:

(REF,X,Y)	; Establish an X and Y Home position.
G70	; Initiate English Programming (inches).
G91	; Initiate Incremental Positioning.
G1 X2. Y2. F200.	; Initiate a Linear Offset Move.
G5 X4. Y0. I2. J2. F100.	; Perform Three Point Interpolation.
M0	; Program Stop.

G5/G15/H5/H15

NOTES:

1) The following code/plane relationships exist:

G5 - 1st plane axes	**H5** - 3rd plane axes
G15 - 2nd plane axes	**H15** - 4th plane axes

2) Helical Interpolation is accomplished by programming two axes to do Circular Interpolation and a third axis to do Linear Interpolation.

3) Spherical Interpolation is accomplished by programming two axes to do Circular Interpolation and then programming the third and a hidden fourth axis to also do Circular Interpolation.

4) To indicate the axes involved in the Circular Interpolation, enter a **G17/H17, G18/H18, G19/H19,** or **PLNE** command prior to the **G5/G15/H5/H15** command.

5) It is not necessary for both axes involved in a Circular Contour to have the same resolution.

6) **G5/G15/H5/H15** are not modal commands.

7) When transferring from lines to arcs and back again the User must reestablish **G1** mode of operation

   or

   a "circle missed center point" error will result.

RELATED COMMANDS:

G17/H17, G18/H18, G19/H19,
I/J/K/P and **i/j/k/p,**
PLNE

G8

NAME:

G8 - Velocity Profiling (acceleration)

FUNCTION:

The **G8** command is used for Velocity Profiling. During Velocity Profiling, priority is given to attaining and maintaining the programmed Feedrate of the current motion command. This command also permits M-function output on-the-fly, and provides an accuracy of within 1ms of the commanded output.

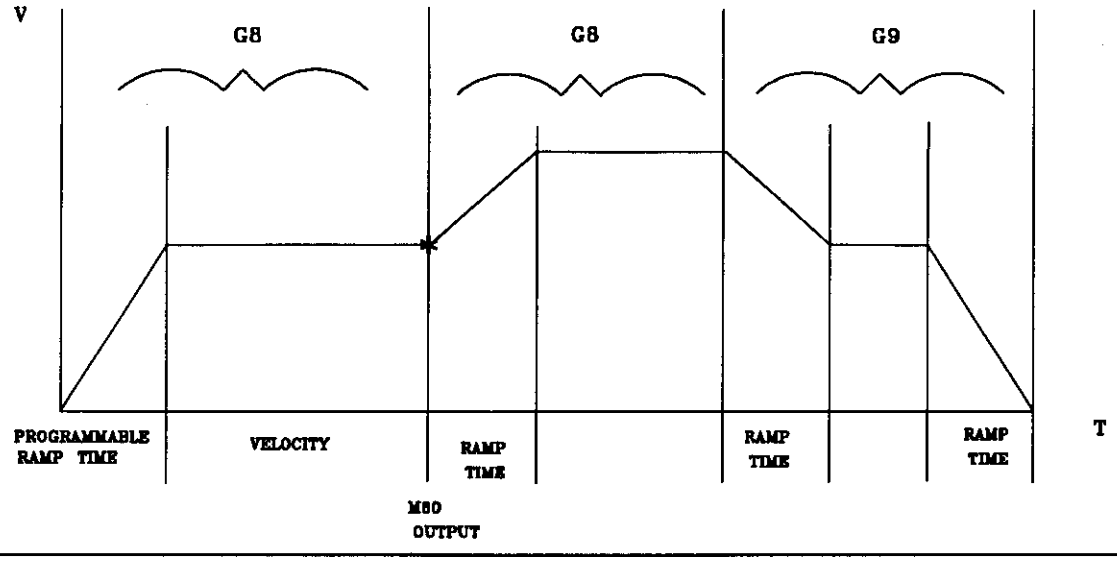
FORMAT:

G8

RETURNS:

EXAMPLE:

```
G24                          ; Use Non-Corner Rounding mode of operation.
G1 G8 X1. F100. M80          ; At the end of X travel, the axis maintains contouring velocity
                               with no deceleration and activates M80 at the end of the 1
                               inch move.
G1 G8 X1. F200.
G1 G9 X1. F100.              ; At the conclusion of G9, move deceleration to zero velocity.
```



G8

NOTES:

1) If the **G8** command is not used, the default provides for both Acceleration and Deceleration to zero velocity.

2) The **G8** command is used for contouring only. If **G0** has been previously programmed the **G8** command will be ignored.

3) The first **G8** command (i.e, after a previously executed **G9** command) will have a beginning vector velocity of zero. Subsequent **G8** commands will be executed beginning at the vector velocity reached by the previous **G8** command.

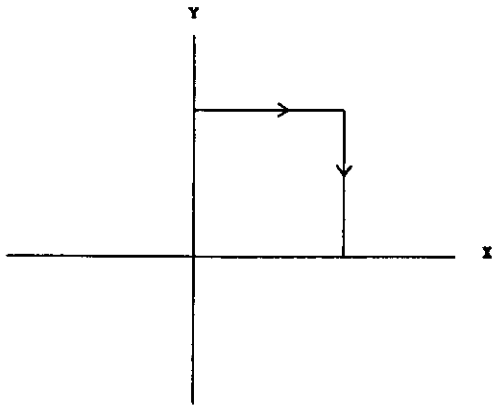
4) The Acceleration/Deceleration mechanism used is dependent upon the current setting of the **G23/G24** modal parameter. (Refer to Appendix 2 "Corner Rounding/Velocity Profiling" for further detail.)

5) To maintain smooth transitions from block-to-block motion, the **G8** command requires that consecutive moves be tangent to each other.

6) Velocity Profiling may be used to produce a non-tangential shape by using arcs to initiate transitional moves. See the following example.

Velocity Profiling **cannot** be maintained smoothly.

G1 **G8** X4.
G1 G9 Y-4.

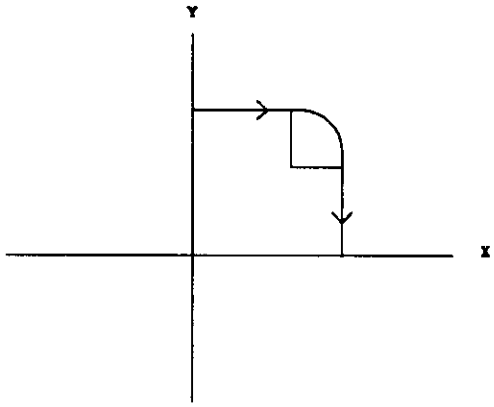


G8

NOTES (CON'T):

Velocity Profiling **will** be maintained smoothly.

G1 **G8** X4.
G2 **G8** X1. Y-1. J-1
G1 G9 Y-4.



RELATED COMMANDS:

COEF, FILT, G9, G23, G24, MST, RAMP, TRAJ

G9

NAME:

G9 - Velocity Profiling (deceleration)

FUNCTION:

The **G9** command is used in conjunction with the **G8** command to provide controlled deceleration to zero velocity after Velocity Profiling. The motion executed under this mode is guaranteed to feed the servo control loop(s) the final position of the specified command.

FORMAT:

G9

RETURNS:

EXAMPLE:

See the example under the **G8** command.

NOTES:

1) A **G9** command not preceded by a **G8** command will have a beginning and ending vector velocity of zero.

2) A **G9** command preceded by a **G8** command has a beginning vector velocity equal to the ending velocity of the previous **G8** command. The ending velocity of the **G9** movement will be zero.

3) The Acceleration/Deceleration mechanism used is dependent upon the current setting of the **G23/G24** modal parameter. (Refer to Appendix 2 "Corner Rounding/Velocity Profiling" for further detail.)

RELATED COMMANDS:

COEF, FILT, G8, G23, G24, RAMP, TRAJ

G17/H17

NAME:

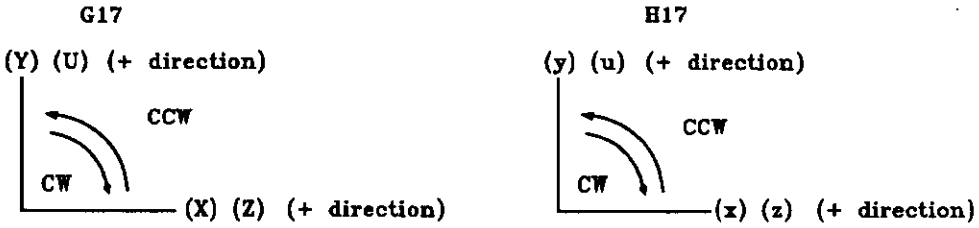
G17/H17 - Axis Plane Designation for execution of Circular Interpolation

FUNCTION:

The **G17/H17** command specifies the axis plane for the execution of Circular Interpolation where:

X/Y axes is the 1st contouring plane (**G1/G2/G3/G5**)
Z/U axes is the 2nd contouring plane (**G11/G12/G13/G15**)
x/y axes is the 3rd contouring plane (**H1/H2/H3/H5**)
z/u axes is the 4th contouring plane (**H11/H12/H13/H15**)

This permits the User to contour between different groups of axes. Refer to the illustration below for **G17/H17** rotational direction.



FORMAT:

G17 or **H17**

RETURNS:

EXAMPLE:

```
G17 G2  X4.0 Y0. I2.0  J0. F100.          ; A CW arc is performed on the XY plane.
```

G17/H17

NOTES:

1) The **G17/H17** command is modal and may be specified once at the beginning of the program or on the same line as the contour. The exception is to utilize Cutter Compensation, where the contour plane must be specified.

2) The **G17/H17** command is a default.

RELATED COMMANDS:

G1/G11/H1/H11, G2 /G12/H2/H12, G3/G13/H3/H13, G5/G15/H5/H15, G18/H18, G19/H19, PLNE

G18/H18

NAME:

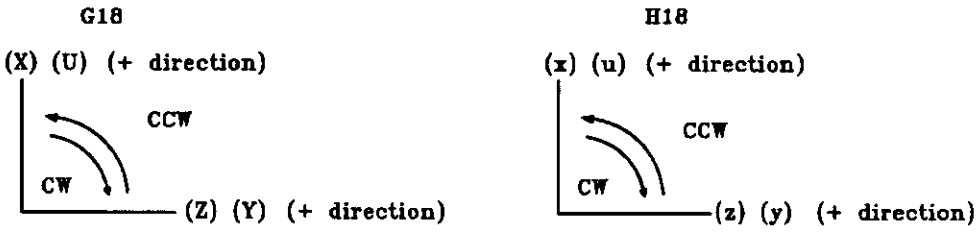
G18/H18 - Axis Plane Designation for execution of Circular Interpolation

FUNCTION:

The **G18/H18** command specifies the axis plane for the execution of Circular Interpolation where:

Z/X axes is the 1st contouring plane (**G1/G2/G3/G5**)
Y/U axes is the 2nd contouring plane (**G11/G12/G13/G15**)
z/x axes is the 3rd contouring plane (**H1/H2/H3/H5**)
y/u axes is the 4th contouring plane (**H11/H12/H13/H15**)

This permits the User to contour between different groups of axes. Refer to the illustration below for **G18/H18** rotational direction.



FORMAT:

G18 or **H18**

RETURNS:

EXAMPLE:

```
G18 G2 Z4.0 X0. K2.0 I0. F100.        ; A CW arc is performed on the ZX plane.
```

G18/H18

NOTES:

1) The **G18/H18** command is modal and may be specified once at the beginning of the program or on the same line as the contour. This exception is to utilize Cutter Compensation, where the contour plane must be specified.

2) The **G17/H17** command is the default.

RELATED COMMANDS:

G1/G11/H1/H11, G2 /G12/H2/H12, G3/G13/H3/H13, G5/G15/H5/H15, G17/H17, G19/H19, PLNE

G19/H19

NAME:

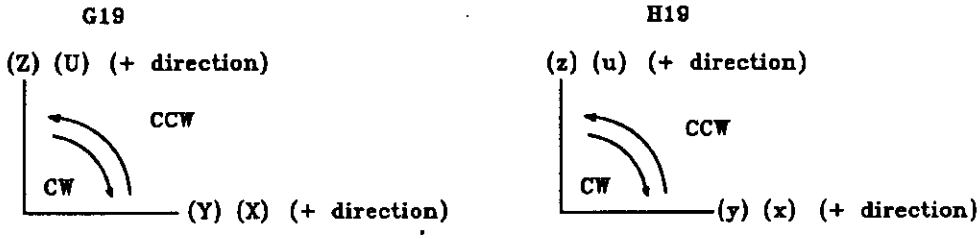
G19/H19 - Axis Plane Designation for execution of Circular Interpolation

FUNCTION:

The **G19/H19** command specifies the axis plane for the execution of Circular Interpolation where:

Y/Z axes is the 1st contouring plane (**G1/G2/G3/G5**)
X/U axes is the 2nd contouring plane (**G11/G12/G13/G15**)
y/z axes is the 3rd contouring plane (**H1/H2/H3/H5**)
x/u axes is the 4th contouring plane (**H11/H12/H13/H15**)

This permits the User to contour between different groups of axes. Refer to the illustration below for **G19/H19** rotational direction.



FORMAT:

G19 or **H19**

RETURNS:

EXAMPLE:

```
G19 G2 Y4.0 Z0. J2.0 K0. F100.        ; A CW arc is performed on the YZ plane.
```

G19/H19

NOTES:

1) The **G19/H19** command is modal and may be specified once at the beginning of the program or on the same line as the contour. The exception is to utilize Cutter Compensation, where the contour plane must be specified.

2) The **G17/H17** command is the default.

RELATED COMMANDS:

G1/G11/H1/H11, G2 /G12/H2/H12, G3/G13/H3/H13, G5/G15/H5/H15, G17/H17, G18/H18, PLNE

G23

NAME:

G23 - Corner Rounding

FUNCTION:

There are two modes of operation for which a motion program may execute. The first ensures that the axes reach their final position before beginning the next motion block, accelerating and decelerating to the end point. This is known as the Non-Corner Rounding Mode (**G24**) and utilizes a Unidex 21 Indexer board "block complete" and "in position" handshake.

The second mode is **G23**, where the system does not wait for the "in position" handshake for the Unidex 21 Indexer board before executing the next block of motion. This is utilized for more traditional machining operations where relatively constant surface speed must be maintained. **G23** is referred to as Corner Rounding due to the effect that is seen when motion commands are executed in sequence prior to the full execution of the previous command.

In both cases, the **RAMP** (Defines Axis Ramping time) and **FILT** (Digital Filter) commands can influence performance and may be utilized in conjunction with the **G23/G24** modes.

The amount of Corner Rounding is influenced by the **RAMP**, **FILT**, and servo loop settings, and, of course, their related parameter settings. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.) **G23** provides the ability to program "non-tangential" moves possibly resulting in smooth motion when used with the proper **FILT/RAMP** settings. This is different from **G24** motion where move tangency considerations should be observed. See the individual above referenced commands for further discussion and explanation.

FORMAT:

G23

RETURNS:

EXAMPLE:

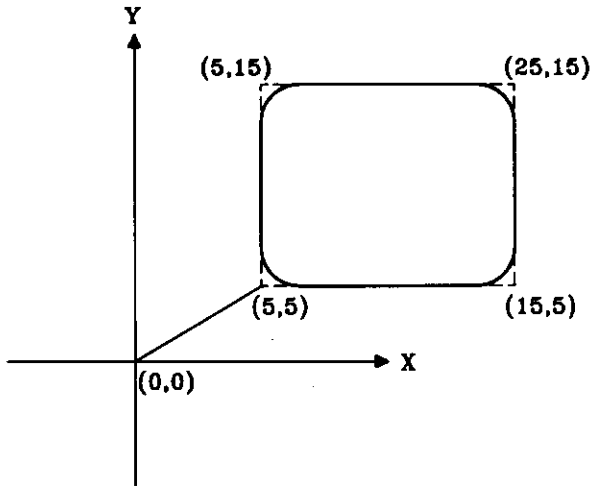
```
G70     ; Initiate English Programming (inches).
G91     ; Initiate Incremental Positioning.
G23     ; Use Corner Rounding mode of operation.
```

G23

EXAMPLE (CON'T):

```
F100.     ; A Feedrate of 100 inches per minute is established.
X10.      ; Move the X axis 10 inches in the positive direction.
Y10.      ; Move the Y axis 10 inches in the positive direction.
X-10.     ; Move the X axis 10 inches in the negative direction.
Y-10.     ; Move the Y axis 10 inches in the negative direction.
```

The above example is illustrated below.



PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #6 located in Appendix 3 of this manual.

NOTES:

1) The **G23** command is modal.

2) The **G24** command is the default.

3) Refer to the Corner Rounding/Velocity Profiling Appendix for further discussion.

RELATED COMMANDS:

FILT, G8, G9, G24, RAMP, SIOC

G24

NAME:

G24 - Non-Corner Rounding

FUNCTION:

There are two modes of operation for which a motion program may execute. The first ensures that the axes reach their final position before beginning the next motion block, accelerating and decelerating to the end point. This is known as the Non-Corner Rounding Mode (**G24**) and utilizes a Unidex 21 Indexer board "block complete" and "in position" handshake.

The second mode is **G23**, where the system does not wait for the "in position" handshake for the Unidex 21 Indexer board before executing the next block of motion. This is utilized for more traditional machining operations where relatively constant surface speed must be maintained. **G23** is referred to as Corner Rounding due to the effect that is seen when motion commands are executed in sequence prior to the full execution of the previous command.

In both cases, the **RAMP** (Defines Axis Ramping time) and **FILT** (Digital Filter) commands can influence performance and may be utilized in conjunction with the **G23/G24** modes.

NOTE: The **FILT** command has no effect on **G24** motion.

The execution of **G24** is influenced by the **RAMP** and servo loop gain settings, and, of course, their related parameter settings. (See the Unidex *21 User's Manual* for a detailed description of the Parameter Mode.) Refer to the **G8** and **G9** commands in addition to the individual above referenced commands for further discussion and explanation.

FORMAT:

G24

RETURNS:

EXAMPLE:

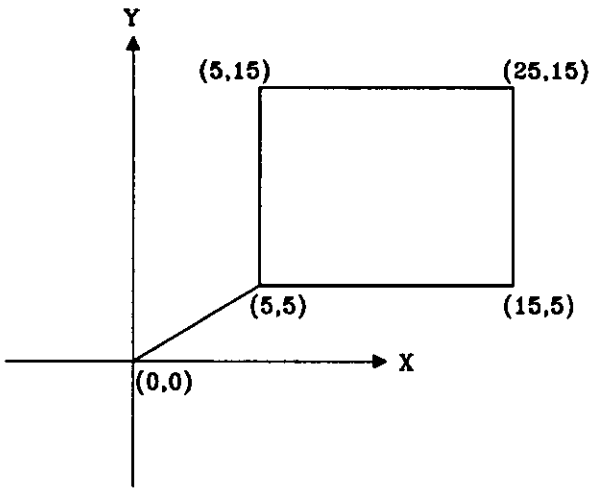
```
G70      ; Initiate English Programming (inches).
G91      ; Initiate Incremental Positioning.
G24      ; Use Non-Corner Rounding mode of operation.
```

G24

EXAMPLE (CON'T):

```
F100.      ; A Feedrate of 100 inches per minute is established.
X10.       ; Move the X axis 10 inches in the positive direction.
Y10.       ; Move the Y axis 10 inches in the positive direction.
X-10.      ; Move the X axis 10 inches in the negative direction.
Y-10.      ; Move the Y axis 10 inches in the negative direction.
```

The above example is illustrated below.



PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #2 and 6 located in Appendix 3 of this manual.

NOTES:

1) The **G24** command is modal.

2) The **G24** command is the default.

3) Refer to the Corner Rounding/Velocity Profiling Appendix for further discussion.

RELATED COMMANDS:

FILT, G8, G9, G23, RAMP, SIOC

G40

NAME:

G40 - Deactivate Cutter Radius Compensation

FUNCTION:

The **G40** command deactivates Cutter Radius Compensation.

FORMAT:

G40 and end move

RETURNS:

EXAMPLE:

```
G40 X1. Y1. F100.        ; Deactivate the Cutter Radius Compensation and perform the
                           end move.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #5 located in Appendix 3 of this manual.

NOTES:

1) The **G40** command is modal.

2) The **G40** command is the default.

3) The **G40** command must be followed by an end move.

4) For more complete examples regarding the use of **G40** with **CCP**, **G41**, and **G42** refer to the Cutter Compensation Appendix.

RELATED COMMANDS:

CCP, G41, G42

G41

NAME:

G41 - Activate Cutter Radius Compensation - Left

FUNCTION:

The **G41** command initiates Cutter Radius Compensation left. The cutter is offset to the left side of the work piece. Left/right is relative to the direction in which the cutter is moving.

FORMAT:

G41

RETURNS:

EXAMPLE:

```
G41 X2. Y2. F100.          ; Activate the Cutter Radius Compensation left.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #5 located in Appendix 3 of this manual.

NOTES:

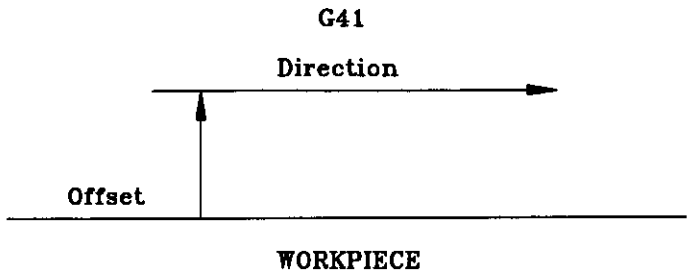
1) The **G41** command is modal.

2) The **G40** command is the default.

3) For more complete examples regarding the use of **G41** with **CCP**, **G40**, and **G42** refer to the Cutter Compensation Appendix.

RELATED COMMANDS:

CCP, G40, G42



G42

NAME:

G42 - Activate Cutter Radius Compensation - Right

FUNCTION:

The **G42** command initiates Cutter Radius Compensation right. The cutter is offset to the right side of the work piece. Left/right is relative to the direction in which the cutter is moving.

FORMAT:

G42

RETURNS:

EXAMPLE:

```
G42 X2. Y2. F100.          ; Activate the Cutter Radius Compensation right.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #5 located in Appendix 3 of this manual.

NOTES:

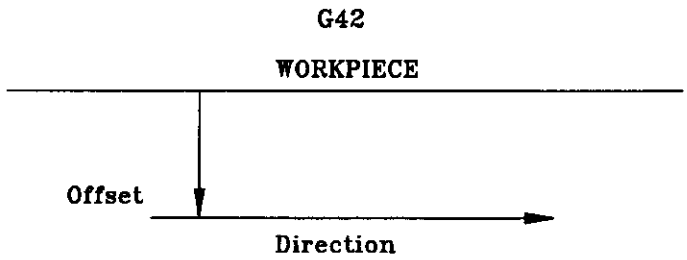
1) The **G42** command is modal.

2) The **G40** command is the default.

3) For more complete examples regarding the use of **G42** with **CCP**, **G40**, and **G41** refer to the Cutter Compensation Appendix.

RELATED COMMANDS:

CCP, G40, G41



G70

NAME:

G70 - English Programming

FUNCTION:

The **G70** command initiates English Programming. All dimensional and offset commands will be represented in inches.

FORMAT:

G70

RETURNS:

EXAMPLE:

```
G70          ; Initiate English Programming (inches).
X10.         ; Move the X axis 10 inches in the positive direction.
Y5.          ; Move the Y axis 5 inches in the positive direction.
F100.        ; A Feedrate of 100 inches per minute for the X and Y axes is
               established.
```

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #1, 2, 5, and 6 located in Appendix 3 of this manual.

NOTES:

1) The **G70** command is modal.

2) The default command is established in the Parameter Mode and is dependent upon the system's mechanical resolution. For a detailed description of the parameters which set up units and default settings, refer to the General Machine Parameter #5, and the Individual Axis Parameters #0, 1, 2, and 3 as established under the 401-408 grouping of the *Unidex 21 User's Manual.*

RELATED COMMANDS:

G71

G71

NAME:

G71 - Metric Programming

FUNCTION:

The **G71** command initiates Metric Programming. All dimensional and offset commands will be represented in millimeters.

FORMAT:

G71

RETURNS:

EXAMPLE:

```
G71     ; Initiate Metric Programming (millimeters).
X4.     ; Move the X axis 4 millimeters in the positive direction.
Y2.     ; Move the Y axis 2 millimeters in the positive direction.
F100.   ; A Feedrate of 100 millimeters per minute for the X and Y axes
          is established.
```

NOTES:

1) The **G71** command is modal.

2) The default command is established in the Parameter Mode and is dependent upon the system's mechanical resolution. For a detailed description of the parameters which set up units and default settings, refer to the General Machine Parameter #5, and the Individual Axis Parameters #0, 1, 2, and 3 as established under the 401-408 grouping of the *Unidex 21 Users Manual.*

RELATED COMMANDS:

G70

G90

NAME:

G90 - Absolute Position Programming

FUNCTION:

The **G90** command initiates programming in Absolute Position values.

FORMAT:

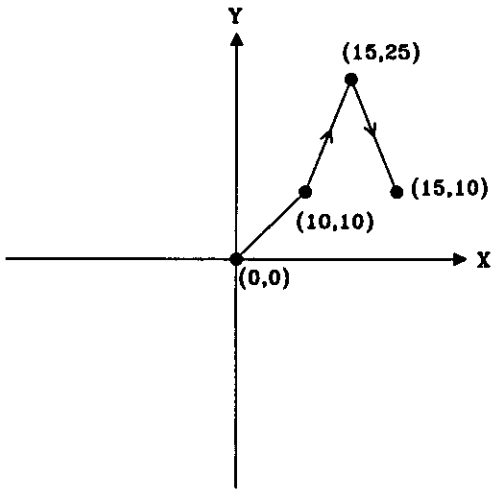
G90

RETURNS:

EXAMPLE:

```
G92 X0. Y0.          ; A Software Register value of X=0, Y=0 is established.
G90                  ; Initiate Absolute Positioning.
X10.0 Y10.0          ; Move the X and Y axes in increments of inches or millimeters
                       from the current position to X=10.0, Y=10.0.
X15.0 Y25.0          ; Move the X and Y axes in increments of inches or millimeters
                       such that the current position is X=15.0, Y=25.0.
X15.0 Y10.0          ; The X axis will not move, the Y axis will move in increments
                       of inches or millimeters to Y=10.0.
```

The above example is illustrated below.



G90

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #1, 2, and 6 located in Appendix 3 of this manual.

NOTES:

1) The **G90** command is modal.

2) While in the **G90** (Absolute mode), if any of the following features are activated, a **G92** (Software Home) position must be reestablished prior to requesting any other moves:

   AFCO, FREE, HWEL, MIR, SCF, SCO, SLEW

3) The **G91** Incremental Positioning mode is the default setting.

RELATED COMMANDS:

$nAP, $nRP, G91, G92

G91

NAME:

G91 - Incremental Position Programming

FUNCTION:

The **G91** command initiates programming in Incremental Position values.

FORMAT:

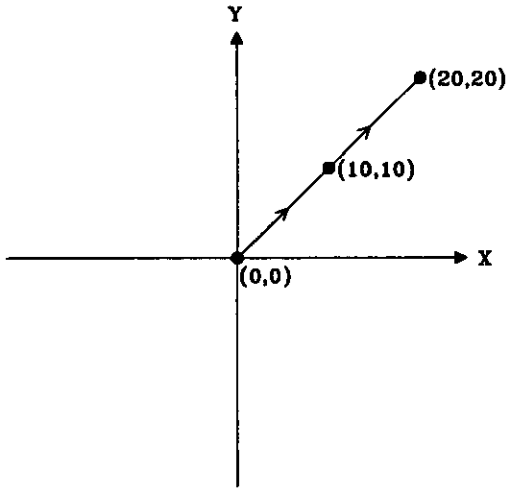
G91

RETURNS:

EXAMPLE:

```
G92 X0. Y0.      ; A Software Register value of X=0, Y=0 is established.
G91              ; Initiate Incremental Positioning.
X10. Y10.        ; Move the X and Y axes 10 inches or millimeters from the
                   current position to X=10.0, Y=10.0.
X10. Y10.        ; Move the X and Y axes 10 inches or millimeters such that the
                   current position is X=20.0, Y=20.0.
```

The above example is illustrated below.



G91

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #1, 2, and 5 located in Appendix 3 of this manual.

NOTES:

1) The **G91** command is modal.

2) The **G91** Incremental Positioning mode is the default command.

RELATED COMMANDS:

$nAP, $nRP, G90, G92

G92

NAME:

G92 - Software Home

FUNCTION:

The **G92** command establishes a Software Register Value for the **$nRP** Machine Position Registers at the current position of either all or the specified axes. The absolute and incremental **G90/G91** programming modes utilize the **$nRP** Position Registers for motion. The **G92** command is useful for setting position offsets, or returning the **$nRP** registers to **$nAP** values. The Machine Position Tracking Display is also affected by the **G92** command.

FORMAT:

G92

or

G92 specified axis name

RETURNS:

EXAMPLES:

G92	; Set all axes to the Software Register value of zero.
G92 X10. Y20.	; Set the Software Registers for the X and Y axes such that the current position for the X axis is 10 inches or millimeters and the current position for the Y axis is 20 inches or millimeters, the remaining axes Software Register values remain unchanged.
G92 X=VAR1, Y=YAP	; Set the X axis $nRP register to the value of variable VAR1, and the Y axis $nRP register to the $YAP value.

NOTES:

1) The **G92** command is not modal.

G92

NOTES (CON'T):

2) The **G92** command changes the value of the **$nRP** Machine Incremental Commanded Position Registers but not the value of the **$nAP** Machine Absolute Commanded Position Registers.

RELATED COMMANDS:

$nAP, $nRP, DVAR, G90, G91

HOME

NAME:

HOME - Hardware Home

FUNCTION:

The **HOME** and **REF** commands are the same. These commands are used to send any or all axes to the Hardware Home position. The axis first moves to the Home Limit Switch, and then moves out until the Home Marker is located. If a Home position is required that is different from the Home Marker position, a Home Offset position may be established within the Individual Axis Parameter settings (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode). The system utilizes two different Home feedrates for homing: Initial power on Home Feedrate or Normal Time Feedrate. This provides for rapid system homing after the Unidex 21's DSP Servo Control card has established the initial Home position. See figures 3-1 and 3-2 for an illustration of axis movement following a Home command.

FORMAT:

(**HOME,** axis name ,axis name,...)

RETURNS:

$nRP and **$nAP** cleared to zero

EXAMPLE:

```
(HOME,X,Y)                    ; The X and Y axes are sent Home simultaneously.
```

ADDITIONAL EXPLANATION:

Figure 3-1 illustrates the use of the Home command following a Power Up in which case the current axis position is unknown. The axis moves toward the Home limit at a constant Feedrate established in the "Power On Home Feedrate, steps/sec" Parameter #28 of the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

Upon reaching the Home Limit Switch, axis direction is reversed and acceleration occurs until the Home Feedrate is reached. This Feedrate is maintained until deceleration (after coming back out of the limit) occurs, completing the Marker Offset distance established in the "Home Limit to Marker, steps" of the Individual Axis Parameters 401-408 grouping. (See also the *Unidex 21 User's Manual* for further detail.)

HOME

ADDITIONAL EXPLANATION (CON'T):

When the Marker Offset distance has been completed, motion will continue at a rate of one-half machine count per millisecond until the Home Marker is reached. Acceleration occurs repeatedly until the Home Feedrate is reached. This Feedrate is maintained until deceleration occurs, completing the Home Offset distance established in Parameter #7 of the Individual Axis Parameters 401-408 grouping, and the Home position is reached. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

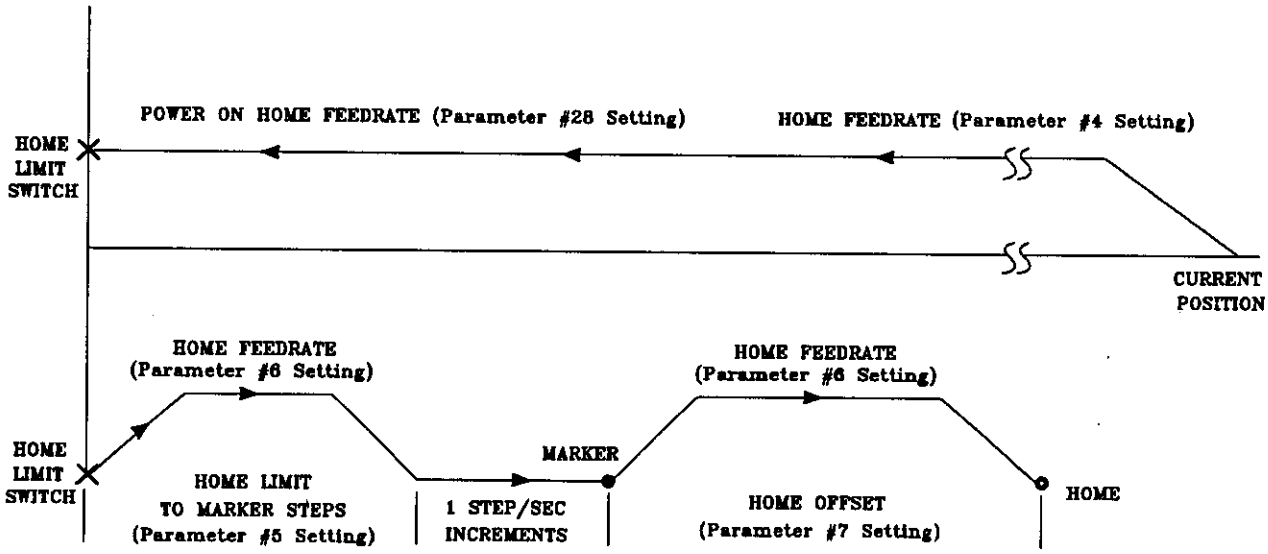


Figure 3-1: Home Command Following a Power Up

Figure 3-2 illustrates the use of a Home command from a known axis position after a Power Up Home command has been executed. The axis approaches the Home position at the Feedrate established in the "Home Feedrate, steps/sec" Parameter #6 of the Individual Axis Parameters 401-408 grouping. (See also the *Unidex 21 User's Manual* for further information.) Deceleration occurs as the axis approaches the Home Limit Switch. (The deceleration rate is 25% of the Individual Axis Parameter Setting-Ac/De,steps/sec^2.) Following deceleration, motion may continue in one step increments until the Home Limit Switch is reached.

Upon reaching the Home Limit Switch, axis direction is reversed and acceleration occurs until the Home Feedrate is reached. This Feedrate is maintained until deceleration occurs, completing the Marker Offset distance established in "Home Limit Marker, steps" parameter of the Individual Axis Parameters 401-408 grouping.

HOME

ADDITIONAL EXPLANATION (CON'T):

When the Marker Offset distance has been completed, motion will continue in one step increments until the Home Marker is reached. Acceleration occurs repeatedly until the Home Feedrate is reached. This Feedrate is maintained until deceleration occurs, completing the Home Offset distance established in the "Home Offset, steps" Parameter #7 of the Individual Axis Parameters 401-408 grouping.

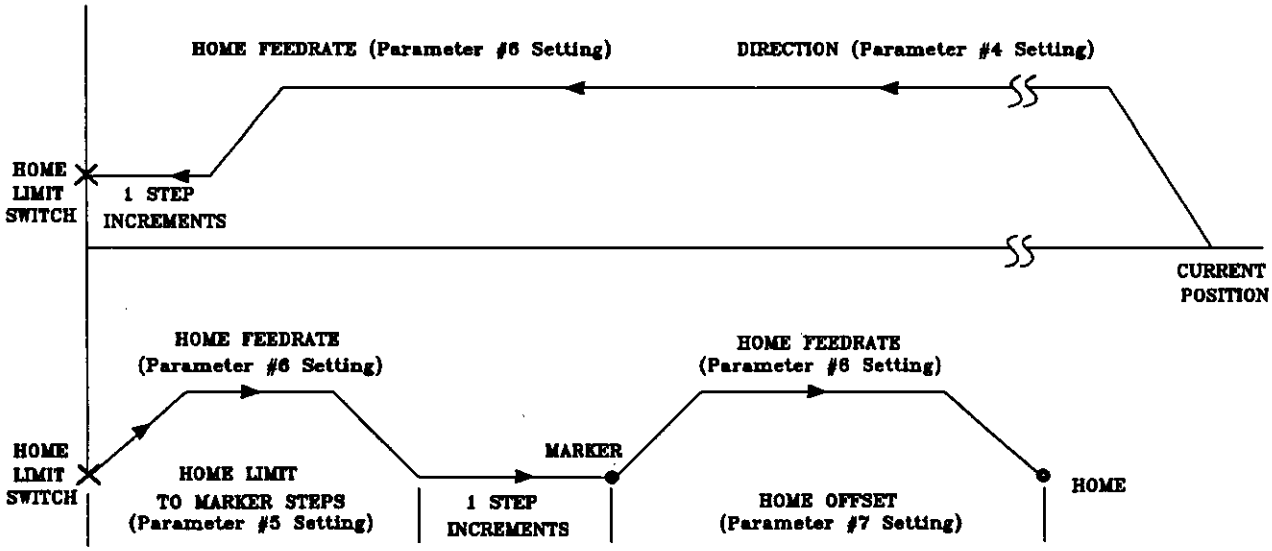


Figure 3-2: Home Command From a Known Position

NOTES:

1) The Home Feedrate may be adjusted without entering the Parameter Mode by using the **MFO** Manual Feed Override.

2) On systems with linear encoders or very fine resolution, it may be necessary to insert a Home Offset or Home Limit to marker steps via Parameter Mode in order to decrease the time for execution of a Home cycle.

3) The applicable parameters are #4, 5, 6, 7, 9, 26, 27, and 28 established in the Individual Axis Parameters 401-408 grouping located in the *Unidex 21 User's Manual.*

HOME

NOTES (CON'T):

4) On rotary systems which have a cam type switch (normally open on one side, and closed on the other) the Unidex 21 will automatically move in the correct direction to the Home position.

RELATED COMMANDS:

$nAP, $nRP, ACDE, FXOF, G92, MORG, REF

HSIE

NAME:

HSIE - High Speed Interrupt Control

FUNCTION:

The **HSIE** command is used for data collection from the CPU card. The data acquisition event is triggered externally, by the User, through the (P23) I/O connector on the Interface panel of the Unidex 21. Data that can be collected consists of system variables, **$nAP**, **$nRP**, **$INP**, **$IN0-$INF**, **$000-$7FF**, or **$800-$F6F**. Typical timing latency for an input device (excluding the system variable) is 45 micro seconds for the first data and 25 micro seconds thereafter unless further latency is encountered due to the read cycle of the data card. The accuracy of the **$nAP** and **$nRP** Machine Commanded Position Register acquisition is dependent upon Unidex 21's ability to access the Indexer board.

FORMAT:

(**HSIE**,option)

RETURNS:

EXAMPLE:

(**HSIE**,1)

PROGRAMMING EXAMPLE:

The following High Speed Interrupt example shows how the **HSIE** command is used to collect data in the background and store that data to a file. This example saves both Absolute and Relative Position to the TEST.DAT file.

```
(DVAR,CNT)                  ; CNT is the variable used to write the data to a file.
G17                         ; The XY plane is the 1st plane.
G40                         ; Ensure Cutter Compensation is OFF.
G70                         ; Initiate English Programming (inches).
G91                         ; Use Incremental Positioning.
F1000.                      ; A Feedrate of 1000 inches per minute is established.
(MALC,<0,200,$XRP,$XAP>)    ; Allocate memory for High Speed Interrupt for a total
                            ; of 200 data items.
```

HSIE

PROGRAMMING EXAMPLE (CON'T):

```
                          ; Record 100 X Axis Incremental Positions,
                          ; and 100 X Axis Absolute Positions.
(HSIE,1)                  ; Enable the High Speed Interrupt.
                          ; Axis position will be captured upon
                          ; each falling edge of the HSI Misc I/O pin.
G1  X100.                 ; Linear Contouring at Feedrate of 100 inches per minute.
X-50.                     ; Move the X axis 50 inches in the negative direction.
X30.                      ; Move the X axis 30 inches in the positive direction.
X-80.                     ; Move the X axis 80 inches in the negative direction.
(HSIE,0)                  ; Disable the High Speed Interrupt and store the
                          ; accumulated data to a file.
(OPEN,W,TEST.DAT)         ; Open the Output file.
CNT=1.                    ; Start with the 1st new data item.
(RPT,$HSI<0>              ; Repeat the number of data items.
  (WRIT,#$HSI<CNT>)       ; Write the data item to the file.
  CNT=CNT+1.              ; Move to the next data item.
)                         ; End of Repeat Loop
(END,S)                   ; Close the File and save all data written.
M2                        ; End of The Program
```

NOTES:

1) Prior to enabling the High Speed Interrupt, the MALC <.....> command must be used to allocate appropriate memory space and specify the data to be captured.

2) The following options are available when using the **HSIE** command:

   option 0 – Disable the High Speed Interrupt.

   1 – Input starts at the same memory location. New data will overwrite previous data.

   2 – New data will start at the next memory location. Data collection will stop when the end of memory is reached.

HSIE

NOTES (CON'T):

option 3 – New data will start at the next memory location. Data collection will stop when the end of memory is reached. When the **HSIE** is enabled, the interface pin on the User I/O bus will go HIGH. When the end of memory is reached the pin goes LOW. (The User may use this function to control an external device or to connect to **INT1/INT2.**)

3) Options 1, 2, and 3 enable the High Speed Interrupt. The function remains in effect and continues to run in the background until the system is reset or the command (**HSIE**,0) is entered, to disable it.

4) All User Interrupt Signals must be debounced. Reference the *Unidex 21's Hardware Manual* for a description of the miscellaneous I/O connector P23.

RELATED COMMANDS:

$HSI, END, INT1/INT2, MALC, MEND, OPEN, RETP, WRIT

HWEL

NAME:

HWEL - Handwheel Control

FUNCTION:

The **HWEL** command initiates Handwheel Control of a designated axis or axes. Two Handwheel/Axis combinations may be used simultaneously, if both inputs are utilized. A Scaling Factor establishes the ratio between Handwheel Steps and Machine Steps. Setting the Scaling Factor to zero disables the Handwheel Control. One Handwheel input may be assigned to only 1 axis at a time.

FORMAT:

(**HWEL**,input source, scaling factor,axis name)

RETURNS:

EXAMPLE:

```
(HWEL,1,50,X)          ; The X axis is activated for Handwheel Control through Input
                         Port 1. Each Handwheel Step equals 50 Machine Steps.

(HWEL,VAR1,0,X)        ; Handwheel Control of the X axis is disabled for Input 1 or 2,
                         dependent upon the value of variable VAR1.
```

NOTES:

1) The input source may be 1 or 2, and may be specified as a variable.

2) The Scaling Factor may be 0 to 255, and may be specified as a variable.

3) The Handwheel inputs may be configured to accept Clock/Direction, CCW/CW clock, Quadrature (X1) or Quadrature (X2). Refer to the General Machine Parameters #28, 29, 42, 43, and 45 located in the *Unidex 21 User's Manual.*

4) Manual Feedrate Override **MFO** may also be utilized to adjust the Handwheel scale. Refer to General Machine Parameter #55, also located in the *Unidex 21 User's Manual.*

RELATED COMMANDS:

AFCO, DVAR

I,J,K,P and i,j,k,p

NAME:

I, J, K, P and **i, j, k, p** - Circular Interpolation parameters

FUNCTION:

The **I, J, K, P** and **i, j, k, p** commands are Circular Interpolation parameters that are parallel to the X, Y, Z, U or x, y, z, u axes respectively. The values may be positive or negative.

When performing Circular Interpolation using the **G2/G12/H2/H12** or **G3/G13/H3/H13** commands, the **I, J, K, P** and **i, j, k, p** commands indicate the center of the arc (radius). These codes program the offset vector, which is the signed incremental distance from the beginning of the arc to the arc center. The offset vector is determined incrementally regardless of whether the Unidex 21 is in the Absolute (**G90**) or Incremental (**G91**) mode.

The offsets for each axis are as follows:

AXIS	OFFSET
X	I
Y	J
Z	K
U	P
x	i
y	j
z	k
u	p

When performing Circular Interpolation using the **G5/G15/H5/H15** commands, the **I, J, K, P** and **i, j, k, p** commands indicate points along the arc. (Any point may be used except for the starting and ending point.)

The Unidex 21 can be programmed to move in a complete circle by one command block that specifies the direction, centerpoint, and axes plane.

I,J,K,P and i,j,k,p

FORMAT:

Inn

or

I = variable

RETURNS:

EXAMPLE:

G17 G2 X2.0 Y-0.5 I1.0 J-0.1	; A CW arc will be produced from the initial position to the X=2.0, Y=-0.5 coordinate position with the center point I=1.0, J=-0.1. See Figure 3-3 below.

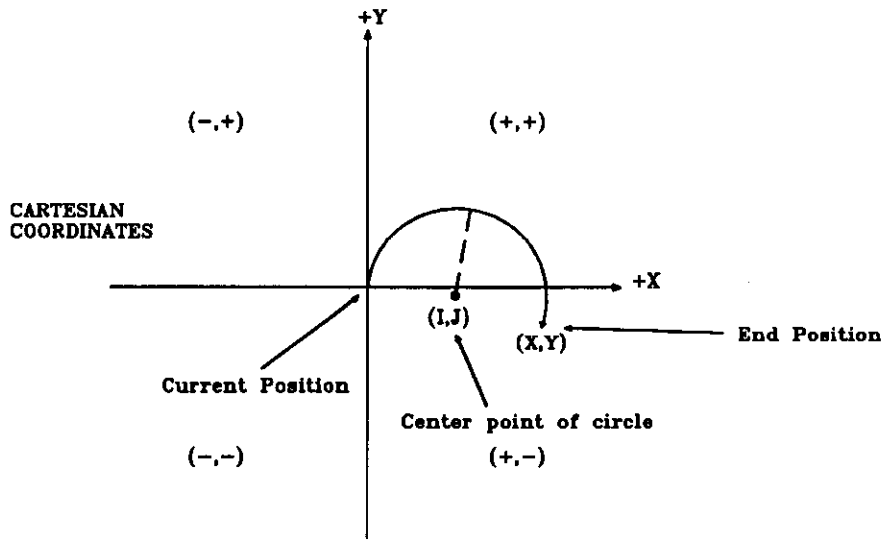


Figure 3-3: Circular Interpolation Illustrating the Use of "I" and "J" as Center Points of an ARC.

I,J,K,P and i,j,k,p

EXAMPLE (CON'T):

G2 G17 I1.0 J1.0	; A CW circle on the X/Y plane will be produced. The ending point is assumed to be 0 (the same as the starting point). The center point is established by the offsets of I=1.0, J=1.0 from the starting point (0,0). See Figure 3-4 below.

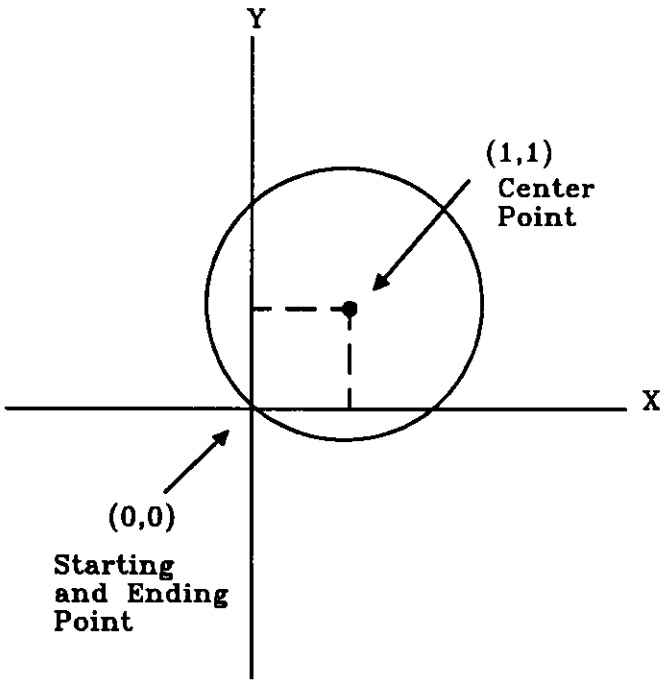


Figure 3-4: Circular Interpolation Illustrating the Use of "I" and "J" for Center Points of a Circle.

I,J,K,P and i,j,k,p

EXAMPLE (CON'T):

G17 G5 X4.0 Y0. I2.0 J2.0	; A CW arc on the X/Y plane will be produced from the initial position to the X=4.0, Y=0. coordinate position crossing points I=2.0, J=2.0. See Figure 3-5 below.

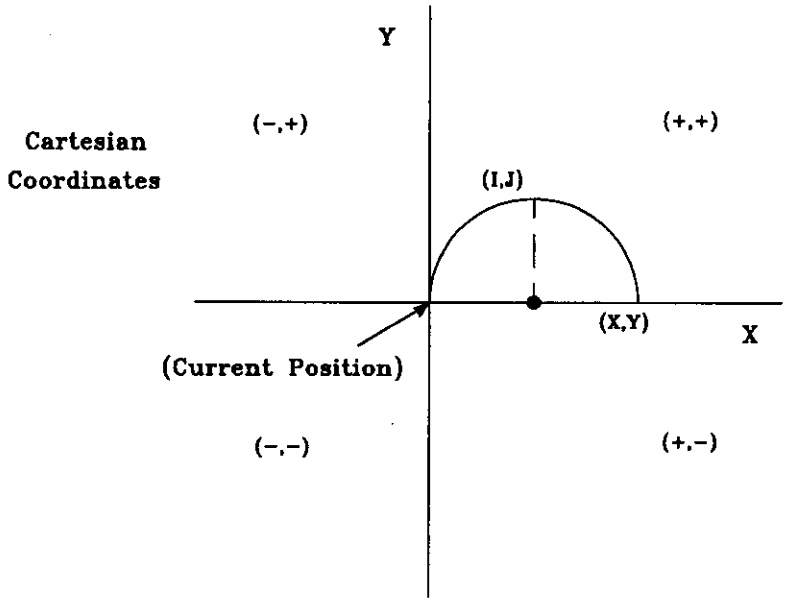


Figure 3-5: Circular Interpolation Illustrating the Use of "I" and "J" as Crossing Points on an Arc.

NOTES:

English/Metric units and decimal point placement used for the **I,J,K,P** and **i,j,k,p** commands will be the same as that established for the corresponding X, Y, Z, U and x, y, z,u axes.

RELATED COMMANDS:

G2/G12/H2/H12, G3/G13/H3/H13, G5/G15/H5/H15, G17/H17, G18/H18, G19/H19, PLNE

INT1/INT2

NAME:

INT1/INT2 - User Interrupt Control

FUNCTION:

The **INT1/INT2** commands are a very powerful feature of Unidex 21 used for permitting instantaneous (7millisecond max latency) response to an input signal in order to redirect program flow. This may be utilized for interlocks or PLC interface.

INT1 and **INT2** are negative edge triggered interrupts, i.e., the interrupt can only be generated when the signal goes from HIGH to LOW. The User may program any one of these interrupts to perform specific interrupt functions. These signals are available via the connectors located on Unidex 21's rear panel Interface board. Similarly, the User may program the system function key in the Parameter Mode in order to emulate this function for **INT1** and **INT2**. Additional Interrupts (**INT3-7**) are also available, see notes below for further explanation.

FORMAT:

(**INT1**,option,xxxx)

The following options are available when using the **INT1/INT2** command (The "xxxx" provides the Interrupt service entry point or Subroutine):

option 0 – Disable Interrupt

1 – Enable Interrupt. Unidex 21 will not stop or abort the current move when jumping to Subroutine xxxx . It will perform defined functions (I/O, mathematical, system variable Input/Output, or MST functions) as long as these are not axis moves. Upon completion of the xxxx Subroutine, the Unidex 21 continues with the next block of functions.

2 – Enable Interrupt. Unidex 21 will stop the current move and abort any functions remaining in this block and store the current machine position. Upon completion of the xxxx Subroutine, the Unidex 21 will return to the next block of program.

INT1/INT2

FORMAT (CON'T):

option 3 – Enable Interrupt. Unidex 21 will stop the current move and abort any functions remaining in this block storing the current machine position. It will then jump to the Defined Entry Point xxxx, but will not precede to the next block when finished. When this option is used in conjunction with the **MSTD** command the User may dedicate **MST** data to be output at the same moment that motion stops. The Unidex 21 does look ahead while interpreting the Parts Program.

4 – Enable Interrupt. Unidex 21 will finish all of the functions in the current block before going to Subroutine xxxx. Upon completion of the Subroutine, the Unidex 21 will return to the next block of program.

5 – Enable Interrupt. Unidex 21 will finish all of the functions in the current block, then jump to the Defined Entry Point xxxx. Upon completion, the Unidex 21 will not return to the next block of program.

6 – Same as option three, but the Unidex 21 does not look ahead while interpreting the Parts Program.

RETURNS:

EXAMPLE:

```
(INT1,1,OFF)          ; Enable Interrupt 1, option 1. ( Subroutine OFF)
X10. Y10. F100.       ; An X and Y axis motion at a Feedrate of 100 inches or
                        millimeters per minute is established.
M2                    ; End of The Program
(DFS,OFF              ; Start of the interrupt service routine.
  (INT1,0)            ; Disable Interrupt 1 to prevent multiple triggers.
  M80                 ;
  (INT1,1,OFF)        ; Re-enable Interrupt 1, option 1. ( Subroutine OFF)
)                     ; End of interrupt service routine.
```

INT1/INT2

NOTES:

1) Once serviced, interrupts (**INT1/INT2**) remain enabled until disabled by the program. This is also true for additional interrupts (**INT3-7**).

2) When Options 2, 4, 5, or 6 are enabled the Unidex 21 does not look ahead while interpreting the Parts Program, thus making program execution slower than Options 1 and 3.

3) You may enable a "dummy" interrupt function with n= 2, 4, or 5 in order to stop Unidex 21 from looking ahead. This is useful when you want to enable the **SKEY** Softkey function but do not want Unidex 21 to look ahead, thus ensuring no blocks of the program are skipped over.

4) All User Interrupt signals must be debounced.

5) The additional interrupts **INT3-6** require usage of the optional Unidex 21 MPI Hardware card. These are functionally similar to **INT1/INT2** except option 3 is unavailable.

6) **INT7** provides a special **MST** output from a hardware signal input to the Unidex 21 either from a special Indexer board pin marked "feedhold" or from the **T** strobe (requires hardware modification).

   The first case format is:

   (INT7,1,Mxx,Sxx,Txx,Mxx) ; Interrupt source is from MST bus connector pin marked "feedhold". Upon each HIGH to LOW trigger the Unidex 21 will output MST data, at most 4 sets. If subsequent interrupts occur before the MST data is executed, they will be ignored.

   The second case format is:

   (INT7,n,Mxx,Sxx,Mxx,Mxx); After MST bus "T" strobe line receives "n" (n=2 to 1,677,215) times trigger MS output will occur, at most 4 sets, with no T data permitted. This requires Indexer board hardware modification and factory initialization.

INT1/INT2

PROGRAMMING EXAMPLE:

This Interrupt example shows how to grab the value found at Input address $700 (exact time an interrupt occurs). These values are placed in an array.

```
G17                     ; The XY plane is the 1st plane.
G40                     ; Ensure Cutter Compensation is OFF.
G70                     ; Initiate English Programming (inches).
G91                     ; Initiate Incremental Positioning.
F1000.                  ; A Feedrate of 1000 inches per minute is established.
(DARY,ARY<100>)         ; Array to hold the values read.
(DVAR,INDX)             ; Array Index is Initialized to 0.
(INT1,1,ISR1)           ; Enable Interrupt INT1.
                        ;
program block           ; These blocks may contain whatever commands
program block           ; are necessary.  Upon receiving an interrupt,
program block           ; the Subroutine ISR1 will be called.  Motion will
program block           ; not be stopped.  When this routine finishes,
program block           ; program execution will continue.
                        ;
(INT1,0)                ; Disable Interrupt INT1.
program block           ; Analyze data placed into the array.
program block           ;
                        ;
M2                      ; End of The Program
(DFS,ISR1               ; Start of the interrupt service routine.
 ARY<INDX>=$700         ; Read the current input value into the array.
 INDX=INDX+1            ; Move to the next array location.
)                       ; End of the interrupt service routine.
```

Also, see Programming Example #4 located in Appendix 3 of this manual for further information regarding use of the **INT1/INT2** commands.

RELATED COMMANDS:

HSIE, MSTD, SKEY

JOIN

NAME:

JOIN - Join Subprogram to Main Program

FUNCTION:

The **JOIN** command is useful when the User wants to separate files which relate to common machine functions, and recall them easily without having to rewrite each within the main Parts Program. Using this command will conserve User RAM. There are two cases under which the **JOIN** command works.

case 1 – applies to a single program which is joined with a Main Program.

2 – applies to the joining of multiple or all programs in memory.

In both cases, the only way for the Main Program to interface with each other is via the **JUMP/CLS** commands. See notes below regarding case 1 and case 2 special considerations.

FORMAT:

(**JOIN**,filename.type)	; Join 1 Subprogram only
(**JOIN**,+,file1.type,file2.type,file3.type, etc...)	; Join multiple subprograms together
(**JOIN**,+)	; Join all programs in memory together

RETURNS:

EXAMPLE:

(**JOIN**,LASER.PP)	; The Subprogram LASER.PP will be joined to the current Main Program.
(**JOIN**,+,LASER.PP,GAS,SHUTTER)	; The selected programs LASER.PP, GAS, and SHUTTER will be joined with the Main Program.
(CLS,GAS)	; Call the GAS Parts Program as a Subroutine.

JOIN

NOTES:

1) The following two cases should be considered when joining a program(s):

case 1 – Single program join

1) The only way for two programs to interface with each other is via **JUMP/CLS.**

2) The following commands need memory management, so they may only be utilized in the Main Program.

END, JOIN, MALC, MEND, JOIN, WRIT

3) The following commands are permitted within all programs and can be accessed by either the Main Program or the Subprogram.

CLS, DARY, DENT, DFLS, DFS, DVAR, DZON, JUMP

case 2 – Multiple program join

1) When multiple files are joined, the Unidex 21 system creates a special $$$.$$$ program for run mode operation only. In this case, each Subprogram will be assigned as a Subroutine, with the first 4 characters of the file name assigned as the Subroutine name. Therefore, these joined programs must follow the rules for variables. i.e., first two characters must be A-Z.

2) Since each Subprogram is called via its first 4 characters it is not possible to utilize **DFLS** or **DFS** as the first lines within the program. Also, each one of the subprograms cannot contain the following:

M2, M30, M47

RELATED COMMANDS:

CLS, DARY, DENT, DFLS, DFS, DVAR, DZON, JUMP

JUMP

NAME:

JUMP - Jump to User Defined Entry Block

FUNCTION:

The **JUMP** command causes program execution to jump to and continue from a specific block. The block entry point is identified by a name which is 2 to 4 characters in length with the restriction that the first two must be letters, A-Z, while the remaining can be any alphanumeric characters.

The **JUMP** command may be one of four types; direct, indirect, conditional direct, or conditional indirect. Examples of each type are provided below.

If a variable is utilized it must be identified with the **DVAR** Define Variable command.

FORMAT:

(**JUMP**,entry point)

RETURNS:

EXAMPLE:

```
(JUMP,AA11)                    ; Program execution goes directly to Entry
                                 Point AA11.

(JUMP,#VAR1)                   ; Program execution goes to a Defined Entry
                                 Point. Variable VAR1 contains the ASCII
                                 name of the entry point.

(JUMP,ENT1,VARA.EQ.VARB)       ; Program execution goes to Entry Point
                                 ENT1 if variable VARA =VARB.

(JUMP,#VAR1,VARA.NE.VARB)      ; Program execution goes to the entry point
                                 contained in variable VAR1 if VARA does not
                                 equal VARB.

(JUMP,END,$IN0.EQ.H,00)        ; Program execution goes to Entry Point END
                                 if input 0 is zero (ON).
```

JUMP

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #1, 2, 3, and 7 located in Appendix 3 of this manual.

NOTES:

1) Do **NOT** duplicate names when defining Jumps and Subroutines.

2) The (**JUMP**,nnnn) command should occupy its own program block.

RELATED COMMANDS:

CLS, DENT, DVAR, INT1/INT2

LCD, OAB and lcd, oab

NAME:

LCD, OAB, and **lcd, oab** - Polar coordinates for Circular Interpolation

FUNCTION:

The **LCD, OAB,** and **lcd, oab** commands are the Polar commands for Circular Interpolation. They are defined as follows:

LCD	the 1st axes plane
OAB	the 2nd axes plane
lcd	the 3rd axes plane
oab	the 4th axes plane

where;

L,O,l,o	the radius of the circle
C,A,c,a	the starting angle as measured from the Horizontal axes, 0° direction, CCW is positive
D,B,d,b	the ending angle

See Figure 3-6 for an example of Polar Coordinate Programming.

FORMAT:

Lnn **C**d1 **D**d2

L = variable **C**=variable **D**=variable

Both the starting and ending commands are in degrees. If minutes and seconds of an arc are required, they must be entered as fractions of degrees. Example:

C112+2/60+16/3600

LCD, OAB and lcd, oab

FORMAT (CON'T):

The angle may also be entered as a decimal. Example:

D112.03777

RETURNS:

EXAMPLE:

G2 L1. C140. D45. F50.	; Initiate CW Circular Contouring such that an arc of 95° will be produced where: L1. specifies the radius C140. specifies the angle of the starting point D45 specifies the angle of the ending point See Figure 3-6 below.

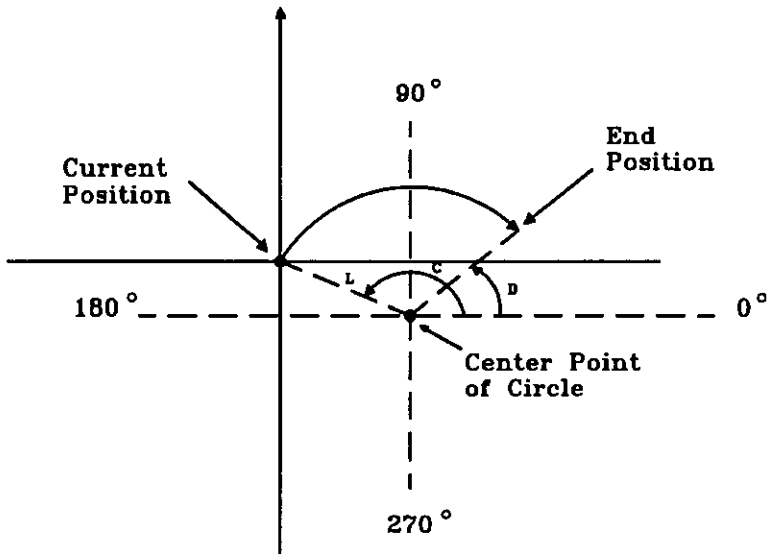


Figure 3-6: Polar Coordinate Programming

LCD, OAB and lcd, oab

PROGRAMMING EXAMPLE:

```
G70                     ; Initiate English Programming (inches).
G91                     ; Initiate Relative Positioning.
(REF,X,Y)               ; A Software Home for the X and Y axes is established.
G1 X2. Y2. F200.        ; Initiate a Linear Move for the X and Y axes,
                        ; at a Feedrate of 200 inches per minute.
G2 L1. C140. D45. F50.  ; Initiate CW Circular Contouring such that an arc of 95°
                        ; will be produced at a Feedrate of 50 inches per minute
                        ; where:
                        ;     L1. specifies the radius
                        ;     C140. specifies the angle of the starting point
                        ;     D45 specifies the angle of the ending point
M0                      ; Program Stop
```

NOTES:

The English/Metric and decimal placement parameters established in the Individual Axis Parameters 401-408 grouping apply to all **L,O** and **I,o** dimensions. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

G2/G12/H2/H12, G3/G13/H3/H13, G17/H17, G18/H18, G19/G19, PLNE

LIMT

NAME:

LIMT - Set Software Limit

FUNCTION:

The **LIMT** command is used to establish a CW and CCW Software Limit as referenced to the Home position. The values established by this command will override the values established in the Individual Axis Parameter Mode, but does not change the Parameter Mode settings. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.) The values established by this command will be in effect until a subsequent **LIMT** command, or the system is reset.

FORMAT:

(**LIMT**, case, axis 1 name CW limit, axis 1 name CCW limit, axis 2 name....)

The cases are:

case 0 – removes any Software Limits of the specified axes.

1 – limits are referenced from the Hardware Home position using Absolute coordinates.

2 – limits are referenced from the current position Incrementally.

RETURNS:

EXAMPLE:

```
(LIMT,0,X0.,Y0.)              ; Disable the X and Y axes Software Limit
                                checking.

(LIMT,1,X20.,X100.,Y20.,Y100.) ; CW and CCW limits of 20 and 100 program
                                units from the Hardware Home position are
                                established for the X and Y axes.

(LIMT,2,X-10.,X10.)           ; Software Limit is set from current X posi-
                                tion, -10.(in/mm) and +10. (in/mm).
```

LIMT

NOTES:

The applicable parameters for setting the CW and CCW Software Limits are #47 and 48 as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of these parameters.)

RELATED COMMANDS:

FXOF, MORG

LINK

NAME:

LINK - Link Multiple Axes to a Group

FUNCTION:

The **LINK** command connects "n" number of axes together in "m" number of groups. Axes that are linked together will perform the same point-to-point or contouring moves.

To retain Tracking integrity for each axis position, all axes linked into the same group should be contained under the same conditions (mirror, scaling, hardware resolution, decimal point placement, dry run, positive move direction, etc.,). However, if the conditions are different use the **G92** (reset software position registers) command after an unlink to reinitialize.

FORMAT:

(**LINK**,<axis name,axis name,...>)

or

(**LINK**,0) ; Un-link all axes.

RETURNS:

EXAMPLE:

```
(LINK,<X,Y,Z>,<U,x>,<y,z>)          ; Put X,Y,Z into the 1st group,
                                    ; U,x into the 2nd group, and
                                    ; y,z into the 3rd group.

X10. U35. y18. F10.                 ; Use linked axis motion.
```

In the above example,

each time the X axis moves, Y and Z will perform the same move.
each time the Y axis moves, X and Z will perform the same move.
each time the Z axis moves, X and Y will perform the same move.

The same is true for groups 2 and 3.

```
(LINK,0)                            ; Un-link all axes.
```

LINK

NOTES:

1) Only one axis should be programmed to move within each group, failure in doing so could result in unpredictable motion.

2) The **LINK** command is modal and remains in effect until changed, or the system is reset.

3) The **LINK** command does not provide a Gantry system, it performs only contouring or point-to-point motion (No Home, Free Run, etc.). To implement Gantry type control, refer to the Main System Parameter #41 located in the *Unidex 21's User's Manual*.

RELATED COMMANDS:

MALC

NAME:

MALC - User's Memory Allocate

FUNCTION:

The **MALC** command is used to Allocate Memory that is to be used for background functions. All functions associated with the **MALC** command run in the background, as a result, the **MALC** command may be input during program initialization. If the **MALC** command is used during a program run, those blocks of memory will become unavailable to the User, and may only be retrieved by overwriting the memory with another **MALC** command or by performing a system reset.

Available background functions are;

<0,n,source1,source2,...>	**HSIE** Memory Allocation. Memory is allocated for n sets of input, where n must be an integer. Each input is 4 bytes in length, where n sets need 4 * n bytes.
<1,n>	**CPAG** Memory Allocation. Memory is allocated for n sets. Information is updated every 200 msec. Each n set requires 20 bytes of memory.
<2,n>	Allocates memory for the **RETP** command to fetch Real-Time position, or for the **RECO** command that is used with the Teach Mode. n specifies the number of 256 byte blocks that are to be allocated for use by this function. Each block is capable of holding 64 position samples.
<3,n>	Allocates memory for the **PORT** command which enables a RS-232 port to collect data. Each n is equal to 1 byte.

MALC

FUNCTION (CON'T):

During the **HSIE** function only, the following system variables are permitted to be used as input sources. Each time an interrupt is received, the data is stored sequentially in memory.

$XRP, $YRP, $ZRP, $URP $XAP, $YAP, $ZAP, $UAP
$xRP, $yRP, $zRP, $uRP $xAP, $yAP, $zAP, $uAP
$INP (16 bits only, so high bits = 0)
$000-7FF (Motorola I/O bus)
$800 -$F6F (PLC DUAL-PORT RAM)
$070 -$FAF PAMUX I/O bus)

FORMAT:

(**MALC**,<background function1>,<background function2>,...)

RETURNS:

EXAMPLE:

```
(MALC,<0,200,$XRP,$INP,$000,$800>)      ; Each time an HSI interrupt occurs, capture
                                          X axis Relative Position, all inputs, Motorola
                                          I/O channel 0, and PLC dual port ram loca-
                                          tion 800.
```

NOTES:

1) Background functions must always be contained within < >.

2) For complete examples on specific uses in each mode, refer to the command activating that mode.

3) The applicable parameter for **MALC** memory options is Parameter #50 established under the Individual Axis Parameters 401-408 grouping (see the Unidex 21 User's Manual for a detailed description of the Parameter Mode).

RELATED COMMANDS:

CPAG, HSIE, PLAY, PORT, RECO, RETP

MIR

NAME:

MIR - Mirror Image

FUNCTION:

The **MIR** command activates the Mirror Image function. The Unidex 21 Mirror Image is available for all eight axes. The **MIR** command operates in both the Incremental and Absolute mode. While in the Absolute Mode however, it must be remembered that functions are always in reference to the Software Home established by the **G92** command which means that you must return to that location before enabling the Mirror Image function.

FORMAT:

(**MIR**,axis name and enable/disable,axis name and enable/disable....)

The Mirror function is enabled by placing a non-zero number next to the axis name.

The Mirror function is disabled by placing a zero next to the axis name.

RETURNS:

EXAMPLE:

```
(MIR,X1.,Y1.)          ; Activate the Mirror Image for the X and Y axes.

(MIR,X0.,Y0.)          ; Deactivate the Mirror Image for the X and Y axes.
```

PROGRAMMING EXAMPLE:

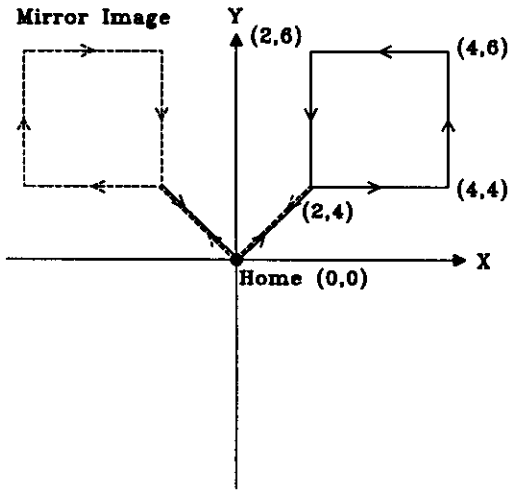
```
G70                    ; Initiate English Programming (inches).
(REF,X,Y)              ; A Software Home for the X and Y axes is established.
(MIR,X1.,Y1.)          ; Activate the Mirror Image function for the X and Y axes,
                       ; changing the positive X and Y values to negative values.
(CLS,BOX1)             ; Call the Subroutine BOX1.
(MIR,X0.,Y0.)          ; Turn OFF the Mirror Image function.
M2                     ; Stop the Program
```

MIR

PROGRAMMING EXAMPLE (CON'T):

```
(DFS,BOX1          ; Define the Subroutine BOX1.
  X2.              ; Initiate a positive Linear Move for the X axis.
  Y4.              ; Initiate a positive Linear Move for the Y axis.
  F100.            ; A Feedrate of 100 inches per minute is established.
  X2.              ; Initiate a positive Linear Move for the X axis.
  Y2.              ; Initiate a positive Linear Move for the Y axis.
  X-2.             ; Initiate a negative Linear Move for the X axis.
  Y-2.             ; Initiate a negative Linear Move for the Y axis.
  X-2.             ; Initiate a Linear Move for the X axis to the Home position.
  Y-4.             ; Initiate a Linear Move for the Y axis to the Home position.
)                  ;
M0                 ; Program Stop
```

The above example is illustrated below.



NOTES:

The **MIR** command is modal.

RELATED COMMANDS:

G92

MORG

NAME:

MORG - Machine Origin

FUNCTION:

The **MORG** command is used to return the axes to a Home Position including the "Machine Origin Offset Steps" Parameter #39 of the Individual Axis Parameters (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode). After performing motion, the User can quickly return to the Home Offset position without going through a complete Home Cycle. This command provides a rapid way to reposition to a known location without rehoming the system and is executed at a **G0** Feedrate.

FORMAT:

(**MORG**,axis name,axis name,...)

RETURNS:

The position register value of the **$nAP** is adjusted to zero, the Machine Origin Offset upon completion.

The **$nRP** Position Register remains the same value as was in effect prior to issuing the **MORG** command.

EXAMPLE:

```
(MORG,X,Y)          ; The X and Y axes are sent to the location established in the
                      Individual Axis Parameter Mode for the X and Y axes.
```

PROGRAMMING EXAMPLE:

```
G70                 ; Initiate English Programming (inches).
G91                 ; Initiate Incremental Positioning.
(REF,X,Y)           ; A Hardware Home for the X and Y axes is established.
(MORG,X,Y)          ; Machine Origin command for the X and Y axes. The X axis
                    ; and the Y axis are offset by the distance specified by their
                    ; Machine Origin Offset (Parameter #39).
M0                  ; Program Stop
```

MORG

NOTES:

1) The applicable parameter is "Machine Origin steps" #39 as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

2) The axes speed is determined by the "Top Feedrate steps/sec" Parameter #8 established under the Individual Axis Parameters 401-408 grouping (see also the *Unidex 21 User's Manual*). The Acceleration/Deceleration Ramping Trajectory is always parabolic.

3) The **MORG** distance is a positive or negative number referenced from the Hardware Home exclusive of any Offsets.

4) Review the **$nAP/$nRP** command description and program example for further explanation.

5) The **MORG** command executes point-to-point motion only.

RELATED COMMANDS:

$nAP, $nRP, FXOF, G0, G92

MSG

NAME:

MSG - Display Screen Messages

FUNCTION:

The **MSG** command permits a message to be entered into a program and displayed on the screen at the time of execution. The message displayed may consist of text plus the value of User variables and system variables. The message may also prompt User input in order to control program flow and set the value of variables. The system defaults to only permitting messages to contain four lines of text. During program execution, message program lines may sometimes be confused with the Parts Program scrolling on the screen. If the (**TRAK**,2) command is utilized, the message area is more readable.

FORMAT:

(**MSG**,...message) ; To display message only

(**MSG**,<VAR1,VAR2>,...message...) ; To display message and accept User input into variables VAR1 and VAR2

The Unidex 21 recognizes the following format for display:

#VAR	as the decimal representation of the variable
#H:VAR	as the Hexadecimal format of a variable
#C:VAR	as the character format
#C:VAR1,VAR2,VAR3	as a variable string

To provide the variable input function the Unidex 21 must be in the Machine Mode.

To output a "#" to the display use "##".

The Message can NOT contain parenthesis (except for the opening and closing parenthesis). Use the "<" and ">" instead.

RETURNS:

MSG

EXAMPLE:

(**MSG**<VAR1,VAR2,VAR3,VAR4>...text...)

if the input data is:

12,H,10FA,"Unidex21"<ENTER>

then,

VAR1 = 12 as a real number
VAR2 = 10FA, as a hexadecimal number
VAR3 = "Unid"
VAR4 = "ex21"

If a longer character string was requested, more **VAR** commands would have been necessary, since a variable holds only four characters. (A space between characters is counted as a character.)

PROGRAMMING EXAMPLE #1:

The following example is designed to show the operation of the **MSG** command when outputting both text and variables.

```
(DVAR,STR1,STR2,FILT,BIN)          ; STR1 and STR2 are to be used for
                                   ; ASCII data.
                                   ; FILT is to hold floating point data.
                                   ; BIN is to hold binary data.
STR1="Test"                        ; Initialize all variables.
STR2="ing"                         ;
FILT=1234.5678                     ;
BIN=H,486121                       ;
                                   ; The message displayed will be:
(MSG,Beginning Test)               ; Beginning test
(MSG,#C:STR1)                      ;    Test
(MSG,#C:STR2)                      ;     ing
```

MSG

PROGRAMMING EXAMPLE #1 (CON'T):

```
                                ; The message displayed will be:
(MSG,#C:STR1,STR2)              ;    Testing
(MSG,Beginning #C:STR1,STR2)    ;    Beginning Testing
                                ;
(MSG,#H:STR1)                   ;    54657374
                                ; Hex representation of ASCII codes.
                                ;
(MSG,#FLT)                      ;    1234.5678
                                ; The Decimal representation of floating point
                                ; format.
(MSG,#H:FLT)                    ;    9A522B4B
                                ; The Hex representation of a floating point
                                ; number.
(MSG,#H,BIN)                    ;    486121
(MSG,#C:BIN)                    ; Ha!
                                ; Character representation of hex data.
M2                              ; End of The Program
```

PROGRAMMING EXAMPLE #2:

The following example shows the operation of the **MSG** command when input is being received form the User.

```
(DVAR,STR1,STR2,FLT,BIN)        ; Variables STR1 and STR2 are to be
                                ; used for ASCII data.
                                ; Variable FLT is to hold floating point data.
                                ; Variable BIN is to hold binary data.
                                ; Prompt the User.
(MSG,<STR1,STR2,FLT,BIN>,
  Enter 5 - 8 Character String,
  a floating, and a binary number
  separated by commas)          ;
M2                              ; End of The Program
```

MSG

PROGRAMMING EXAMPLE #2 (CON'T):

The screen will be displayed as:

"Enter 5-8 Character String, a floating, and a binary number separated by comma's"

USER INPUT	VARIABLE VALUES
"ABCDEFGH",1234.5678,H,55AA	STR1 = ABCD STR2 = EFGH FLT = 1234.5678 BIN = H,55AA
"AB","CD",25.78,H,BE2F	STR1 = AB STR2 = CD FLT = 25.78 BIN = H,BE2F
"Unidex",1965.0419,H,FAC0	STR1 = Unid STR2 = ex FLT = 1965.0419 BIN = H,FACO
"This is a test"	STR1 = This STR2 = is FLT = a te BIN = st

As can be seen from this example, any input received from the User should be checked for validity prior to using it.

Also, for additional information of how this command may be used, refer to Programming Examples #1, 2, 3, 4, and 7 located in Appendix 3 of this manual.

MSG

NOTES:

1) The Unidex 21 will display a real constant greater than 99999999 as:

   1.000000E8

   and a real constant less than 0,0000001 as:

   1.0E-8.

2) The "up arrow" and/or "down arrow" keys may be used to recall previously entered data.

3) Input is terminated when the User presses the ENTER key.

4) There is no type checking of the data being entered by the User. This must be checked in a separate routine which may be part of a User Parts Program.

5) If the User enters less data than expected, unset variables are set to 0.

6) If the User enters more data than can be placed into the variables supplied, unpredictable results may occur.

7) Hex numbers which require more than 4 bytes of storage are truncated to 32 bits. (Least Significant)

RELATED COMMANDS:

$nAP, $nRP, COMM, CPAG, DVAR, TERM

M, S, T

NAME:

M,S,T - Output Capability (16 bit outputs)

FUNCTION:

The Unidex 21 provides additional output capability through the MST bus. Historically, the MST bus has controlled Machine Spindle and Tool Functions. Today, the Unidex 21's MST bus provides output on-the-fly capability when used in conjunction with the **G8** and **G9** commands and can control an analog output through the use of an (SDA) Spindle D to A option card. The **M, S,** or **T** commands are used exclusively as outputs. Each command contains 16 bits, **M0** to **M0FFFF**, **S0** to **S0FFFF**, and **T0** to **T0FFFF** respectively. The duration of the output and the delay time to receive an acknowledgement for **M, S,** and **T** commands is established in the Individual Axis Parameter Mode. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

M commands may be interpreted in two ways by the Unidex 21.

The following **M** commands provide special functions:

M0	Program Stop. Upon completion of all other words in the block, the program will stop. To continue the program, press the CYCLE START pushbutton located on the front panel of the Unidex 21.
M1	Optional (planned) Stop. When the **M1** command is decoded (the OPTIONAL STOP command is active) program execution will stop. To activate the OPTIONAL STOP function, depress the toggle type switch located on the front panel of the Unidex 21. However, if the OPTIONAL STOP function is deactivated, the **M1** command will be ignored by the Unidex 21. To continue running the program, press the CYCLE START pushbutton located on the front panel of the Unidex 21.
M2	End of The Program. After completion of all other commands, program execution will stop. Press the CYCLE START pushbutton on the Unidex 21's front panel to return to the program's beginning and restart the program run.
M30	End of The Program. The program will not run again if CYCLE START is pressed.

M, S, T

FUNCTION (CON'T):

M47 Return to Program Start. When the **M47** command is decoded, program execution will continue from the program's beginning.

Aerotech offers a Latched M-function (LM16) option card (see the *Unidex 21 Options Manual* for further details) as well as provide jumper selectability of addresses. However, when utilizing the LM16 card option, if one of the above system "**M**" codes is needed to serve as an output, it is necessary to program them as a three digit "**M**" function.

For Example:

```
M2      ; Normally signifies the End of The Program, but due to request
          from the User.

M102    ; Permits M2 to be configured as an output. (Aerotech's LM16
          option card decodes only the last two digits.)
```

FORMAT:

Mxxxx, Sxxxx, Txxxx

M=variable, **S**=variable, **T**=variable

The value of the **M, S, T** commands may be either positive or negative.

If the data begins with an A, B, C, D, E, or F, make certain that it is preceded with a "0".

RETURNS:

EXAMPLE:

```
M0011     ; OK
M1AB      ; OK
SABCD     ; Will not be correctly decoded
S0ABCD    ; OK
S-2000    ; OK
M80       ; OK
```

M, S, T

PROGRAMMING EXAMPLE:

For an example of how the **M0**, **M1**, **M30**, and **M47** commands may be used, refer to Programming Examples #1, 2, 3, 4, 6, and 7 located in Appendix 3 of this manual.

NOTES:

The applicable Main System Parameters are #32, 33, 34, 35, 36, and 37 (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode).

RELATED COMMANDS:

G8, G9, MSTD

MSTD

NAME:

MSTD - MST Strobe/Ack Delay, also Output during INT1/INT2, Option 3

FUNCTION:

The **MSTD** command is used to override the **M, S,** and **T** commands Strobe/Ack Delays as established in the Individual Axis Parameter Mode. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.) The Parameter values established by this command will be in effect until a subsequent **MSTD** command, or the system is reset.

When this command is used in conjunction with the **INT1/INT2** Option 3, the User may dedicate **M, S, T** Data to be output at the same moment that motion stops.

As an option, the User may utilize these strobe and acknowledge hardware lines as a handshaking to an external device.

FORMAT:

(**MSTD**,n,mmmm)

The following codes are used for the n and mmmm entries:

n =	0	**M** Strobe Delay	8	**INT1** - S
	1	**M** Ack Delay	9	**INT1** - T
	2	**S** Strobe Delay	10	**INT2** - No MST output
	3	**S** Ack Delay	11	**INT2** - M
	4	**T** Strobe Delay	12	**INT2** - S
	5	**T** Ack Delay	13	**INT2** - T
	6	**INT1** - No MST output		
	7	**INT1** - M		

Note that (6/7/8/9) and (10/11/12/13) are mutually exclusive groups i.e., (that is, n = 6 will override a previous n = 7, 8, or 9).

mmmm = 0 to 65535 delay in msec

or

mmmm = **M, S, T** data

MSTD

RETURNS:

EXAMPLE:

(MSTD,0,100)	; The **M** Strobe Delay is set to 100 msec.
(MSTD,7,80)	; At the moment that **INT1** Option 3 Interrupt stops motion, **M80** will be output.
(MSTD,7,=VAR1)	; The Output data is a variable.

NOTES:

1) If an Ack Delay is equal to zero, the acknowledgment is not checked.

2) If an Ack Delay is set to 65,535 msec, the Unidex 21 will scan indefinitely checking every 1 msec for an acknowledge signal.

3) The applicable Main System Parameters are #32, 33, 34, 35, 36, and 37 (see the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode).

RELATED COMMANDS:

INT1/INT2, M,S,T, MSTD

MTOR

NAME:

MTOR - Motor Current Command Control

FUNCTION:

The **MTOR** command is used to Activate or Deactivate the current command to the servo amplifier and thus the motor torque of an individual axis or all axes. This is useful when the User wishes to disable axes torque and freely move the axes around. Also, during program debug the User may run his actual Parts Program (less motion) in the Machine Mode and utilize the Digital Scope of the Unidex 21 in order to view each axes trajectory commands.

FORMAT:

(**MTOR**,on/off,axis name,axis name,...)

The axis motor is Activated by placing a non-zero number next to the command.

The axis motor is Deactivated by placing a zero next to the command.

RETURNS:

EXAMPLES:

(**MTOR**,0, X,Y,VERT)	; The motor torque to the X, Y, and VERT axes is turned OFF.
(**MTOR**,1,U,x)	; The motor torque to the U and x axes is turned ON.
(**MTOR**,0)	; All axes motor torque is turned OFF.
(**MTOR**,1)	; All axes motor torque is turned ON.
(**MTOR**,VAR1,X,Y)	; The motor torque of X and Y axes is either ON or OFF dependent upon the value of variable VAR1.

MTOR

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #2 located in Appendix 3 of this manual.

NOTES:

1) The **MTOR** command will not activate a motor that has been deactivated by the use of the Emergency Stop pushbutton (Unidex 21 Emergency Stop circuitry option).

2) The **MTOR** command is NOT a substitute for Emergency Stop circuitry. Only the current command to the amplifier is disabled, not the amplifier itself.

3) While the system position tracking display is updated, position of variables **$nAP** and **$nRP** are not updated until the motor torque is restored.

4) Usage on a vertical axes with/without brake control circuitry may cause the axis to fall as the brake is not automatically engaged.

5) The axis traps are defeated when motion torque is turned OFF. The applicable parameters are #29, 32, 33, 34, and 35 as established under the Individual Axis Parameters 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

$nAP, $nRP, DRUN, DVAR

OPEN,MEND,END,WRIT

NAME:

OPEN,MEND,END,WRIT - Reads/Writes data to memory

FUNCTION:

These commands are used to Read and/or Write data to memory. (See examples below.)

The **$R** (Read) command is a system variable that initiates the read of one element of data from a Read file, then increments the pointer to the next available data. However, if retrace is being used the **$R** command will move back one element and read.

FORMAT:

(**OPEN**,R,Filename.type)	; Open a designated file for Read only.
(**OPEN**,W,filename.type)	; Open a designated file for Write only. An error format will result if the file currently exists.
(**OPEN**,w,Filename)	; Open the designated file for Write only. If the file currently exists, the old data will be deleted.
(**MEND**,R)	; Restore the Read pointer to the beginning of the file.
(**MEND**,W)	; Restore the Write pointer to the beginning of the file.
(**END**,S)	; Close the Write file and save the data.
(**END**,A)	; Close the Write file and abort the data.
(**WRIT**,text and data)	; Write the data to the Write file. (format same as **MSG**)

RETURNS:

OPEN,MEND,END,WRIT

EXAMPLE:

```
(DVAR,HEX,FLT,STR1,STR2)        ; Define all variables.
HEX=H,AA55                      ; Initialize each variable to a known state.
FLT=77.69                       ;
STR1="Aero"                     ;
STR2="tech"                     ;
(OPEN,W,TEST.OUT)               ; Open the file "TEST.OUT".
                                ; File Output
(WRIT,#C:STR1,STR2)             ; Aerotech
(WRIT,Data Acquired on #$TOD)   ; Data acquired on 09-JAN-2004  11:31:28
(WRIT,Variable HEX=#H:HEX)      ; HEX = AA55
(WRIT,Variable FLT=#FLT)        ; FLT = 77.69
(END,S)                         ; Close the file "TEST.OUT".
M2                              ; End of The Program
```

Assume the file TEST.IN contains the following data items:

```
1234.5678
H,7F3B
"Read"
"Me"
```

```
(DVAR,HEX,FLT,STR1,STR2)        ; Define all variables.
(OPEN,R,TEST.OUT)               ; Open the file "TEST.OUT".
FLT=$R                          ; FLT = 1234.5678
HEX=$R                          ; HEX=H,7F3B
STR1=$R                         ; STR1 = Read
STR2=$R                         ; STR2 = Me
(MEND,R)                        ; Move the Read pointer to the top of the file.
STR2= $R                        ; STR2 = 1234.5678 (Access as Float)
STR1= $R                        ; STR1 = H,7F3B (Access as Hex)
HEX= $R                         ; HEX = Read (Access as Char)
FLT= $R                         ; FLT = Me (Access as Char)
(END,A)                         ; Close the Read file.
```

OPEN,MEND,END,WRIT

NOTES:

1) If a Write file is opened from the Machine Mode, the system will remain in the Machine Mode until the file is closed (using **END,S**, or **END,A**).

2) The output of the **WRIT** command is in ASCII characters (not binary). Therefore, text can be embedded into the file for readability. It can then be printed on a normal text printer, or edited using the editor of preference.

3) The data items read using the **$R** system variable can be of any type supported by the system (i.e., Float, Hex, or Character). However, the type of the data being read depends upon the type of data found in the file, not the type of the variable it is being read into.

   The following rules apply to the data in the Read file:

   a) Hexadecimal data must be preceded by capital "H," (i.e., H,AA55). Failure to use a capital "H" will result in a undefined variable error.
   b) Character data must be no longer than 4 characters in length and enclosed in quotes (i.e., "HELP").
   c) Floating point data requires no special formatting.

4) When all data has been read from a file, subsequent reads will return zero as the data.

RELATED COMMANDS:

$R, DVAR, MSG

PID

NAME:

PID - Kp, Ki, Kd, Kf1, and **Kf2** Value

FUNCTION:

When peak performance must be maintained, under changing functional, inertial, or velocity conditions, it may be necessary to change the servo gain parameters. The **PID** command was designed to provide the User with the ability to set the **Kp, Ki, Kd, Kf1,** and **Kf2** values for each axis, overriding the Individual Axis Parameter Mode settings. The Parameter values are not changed by this command. The values established by this command will be in effect until a subsequent **PID** command, or the system is reset.

FORMAT:

(**PID**,axis name,gain,and value)

RETURNS:

EXAMPLE:

```
(PID,X,P10.,I=VAR1,D=ARY<3>,F1=20.,F2=30.)   ; The PID F1 and F2 values will be set only for
                                                the X axis.

(PID,Y,D)                                    ; Return the PID values to the default values
                                                (Parameter setting) for only the Y axis.
```

NOTES:

1) Changing the **Kp, Ki, Kd, Kf1,** and/or the **Kf2** values for an axis will affect the motion control response of that axis. Therefore, the values should not be changed during motion.

2) All values for an axis must be specified, even if only changing one item.

3) Only one axis may be changed at a time.

RELATED COMMANDS:

DVAR

PLAY

NAME:

PLAY - Play Back the Recorded Axes Motion

FUNCTION:

The **PLAY** command initiates a repeat of motion that has been recorded using the **RECO** command.

FORMAT:

(PLAY,n)

The following options are used for **PLAY** command entries:

If n = 1 – motion will commence to the first recorded position at **G0** then continue the recorded motion. Tracking Display ON

-1 – motion is the same as 1 above. Tracking Display OFF

2 – motion will commence from the axes current position. Tracking Display ON

-2 – motion is the same as 2 above. Tracking Display OFF

RETURNS:

EXAMPLE:

```
(PLAY,1)          ; Axes will move at G0 to first recorded position, and continue
                    the recorded motion.
```

NOTES:

1) Prior to using the **PLAY** command, make certain the **RECO** command is disabled.

2) To load a recorded file, see "File Mode" in the *Unidex 21 User's Manual.*

RELATED COMMANDS:

RECO

PLC

NAME:

PLC - Programmable Logic Control Interface

FUNCTION:

The Unidex 21 offers an optional PLC (Programmable Logic Controller) which resides on an AT bus Dual-Port RAM interface. The **PLC** command initiates an Interface with the optional Programmable Logic Controller.

FORMAT:

The following codes are used for **PLC** command entries:

```
(PLC,0)                 ; Reset PLC, clear PLC memory
(PLC,R)                 ; Set the PLC to the Run Mode
(PLC,P)                 ; Stop Run Mode, set to Program Mode
(PLC,L,filename.type)   ; Load a Ladder Program to PLC
```

RETURNS:

EXAMPLE:

```
(PLC,L,LASER.LAD)   ; Load a PLC Ladder Program named LASER.LAD from the
                      User's memory to the PLC.
```

NOTES:

1) A **$800** to **$F6F** memory map connects the Unidex 21 to the **PLC** Dual-Port RAM at the 20000H to 2076FH location.

2) Refer to the PLC in the *Unidex 21 Options Manual* for further detail.

RELATED COMMANDS:

$800 to **$F6F**

PLNE

NAME:

PLNE - Define Circular Contour Plane

FUNCTION:

The Unidex 21 provides full contouring capability for any combination of axes. The **PLNE** command establishes Circular Contour Planes among the eight axes. It is normally used to define planes which contain axis pairs that cannot be generated using the **G17/H17**, **G18/H18**, and **G19/H19** commands. It also provides for redefinition of primary axis plane designation such that the User may create a **G2 G17** contouring plane grouping for any arbitrary pair of axes.

FORMAT:

(**PLNE**,p1a1,p1a2,p2a1,p2a2,p3a1,p3a2,p4a1,p4a2)

where

p1a1	; refers to the 1st plane, 1st axis
p1a2	; refers to the 1st plane, 2nd axis
p2a1	; refers to the 2nd plane, 1st axis
p2a2	; refers to the 2nd plane, 2nd axis
p3a1	; refers to the 3rd plane, 1st axis
p3a2	; refers to the 3rd plane, 2nd axis
p4a1	; refers to the 4th plane, 1st axis
p4a2	; refers to the 4th plane, 2nd axis

The sequence in which the axes names appear defines a three dimensional group.

RETURNS

PLNE

EXAMPLE:

PLANES DEFINED	1st	2nd	3rd	4th
G17/H17	X/Y	Z/U	x/y	z/u
(PLNE,X,Y,Z,U,x,y,z,u)	X/Y	Z/U	x/y	z/u
G18/H18	X/Z	Y/U	x/z	y/u
(PLNE,X,Z,Y,U,x,z,y,u)	X/Z	Y/U	x/z	y/u
G19/H19	Y/Z	X/U	y/z	x/u
(PLNE,Y,Z,X,U,y,z,x,u)	Y/Z	X/U	y/z	x/u
(PLNE,X,x,Y,y,Z,z,U,u)	X/x	Y/y	Z/z	U/u
(PLNE,u,z,Z,x,Y,U,X,y)	u/z	Z/x	Y/U	X/y
(PLNE,z,u,x,y,Z,U,X,Y)	z/u	x/y	Z/U	X/Y

NOTES:

1) All axes must be set up even if less than eight are being used.

2) The three dimensional group defined by the sequence in which the axes names appear is used to determine CW/CCW direction for a particular plane. CW/CCW direction for a plane is determined from the perspective of the positive direction of the remaining axis in the three dimensional group. For Example:

   (PLNE,X,Y,Z,U,x,y,z,u)

   defines the X and Y axis to be in the first plane, and X, Y, and Z to be within the same three dimensional group. CW direction for this plane will be defined with respect to positive positions on the Z axis. That is, if the XY plane were a tabletop, and Z positive was in the upward direction, CW direction would be determined looking down onto the table. However, if Z positive was in the downward direction, CW direction would be determined from under the table, looking up.

PLNE

NOTES (CON'T):

Other three dimensional groups defined in this example are:

Z/U/x, x/y/z, and z/u/X

3) The planes set up using this command are modal. That is, they stay in effect until a subsequent **G17/H17**, **G18/H18**, **G19/H19**, or **PLNE** command is executed.

RELATED COMMANDS:

G2, G3, G17/H17, G18/H18, G19/H19

POP

NAME:

POP - Pull Data Out of User's Stack

FUNCTION:

The **POP** command is used to retrieve data from the program stack that was previously stored by the **PUSH** command.

FORMAT:

(**POP**,data2,data1)

Data is retrieved from the data stack in a "last in - first out" basis.

RETURNS:

EXAMPLE:

```
(PUSH,VAR1,VAR2)        ; Data will be stacked as follows:
                                VAR2
                                VAR1

(POP,VAR2,VAR1)         ; Data will be retrieved in the correct order.

(POP,VAR1,VAR2)         ; Data will be swapped between VAR1 and VAR2.
```

For a more complete example see the **PUSH** instruction.

NOTES:

Each data string stored by the **PUSH** command requires 4 bytes of User stack memory. Program stack size (bytes) is established within the Parameter Mode. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

PUSH, STKP

PORT

NAME:

PORT - RS-232 Port Background Data Collection

FUNCTION:

The **PORT** command is used to Activate or Deactivate an RS-232 port for background data collection. Data is collected and stored in the User's memory (see Notes).

FORMAT:

(**PORT**,option)

RETURNS:

EXAMPLE:

```
(PORT,A)        ; Port A will be activated to receive data, but all previously
                  collected data will be erased.

(PORT,B)        ; Port B will be activated to receive data, but all previously
                  collected data will be erased.

(PORT,0)        ; Any previously activated port will be de-activated.
```

PROGRAMMING EXAMPLE:

This RS-232 **PORT** program example shows how the **PORT** command can be used to permit serial communications. For generality, assume that the program is to wait until a "1" is received, and move the X axis 10 inches in the positive direction. If a "2" is received, the program is to move the X axis 10 inches in the negative direction. If a "3" is received, the program should be terminated.

```
(DVAR,CNT,TMP)  ; CNT determines the number of characters
                ; processed.
                ; TMP is used for temporary storage.
G1              ; All moves will be linear.
G17             ; The XY plane is the 1st plane.
```

PORT

PROGRAMMING EXAMPLE (CON'T):

```
G40                                  ; Ensure Cutter Compensation is OFF.
G70                                  ; Initiate English Programming (inches).
G91                                  ; Initiate Incremental Positioning.
F1000.                               ; A Feed Rate of 1000 inches per minute
                                     ; is established.
CNT=0.                               ; No Characters are processed at this time.
(MALC,<3,100>)                       ; Allocate the receive buffer for 100 bytes.
(PORT,A)                             ; Open RS-232, Port A.
(DENT,GETC)                          ; Poll for characters to be received.
(JUMP,CHK,CNT.NE.$POT<0>)            ; If character is received, jump to CHK
(JUMP,GETC)                          ; otherwise, continue to look for it.
(DENT,CHK)                           ; Characters received, check if good.
CNT=CNT+ 1.                          ; Adjust characters processed count.
TMP=CNT                              ; Use TMP to hold the array index.
(JUMP,CHK2,CNT.LT.100)               ; Jump to CHK2 if not the last entry of array.
                                     ; Allocated memory is full.
(PORT,A)                             ; Turn ON again to reset the Pointer.
CNT=0.                               ; Reset the count.
(DENT,CHK2)                          ; Check for "1","2","3".
(JUMP,RX1,$POT<TMP>.EQ.H,31)         ; Jump to RX1 if byte received is a "1",
(JUMP,RX2,$POT<TMP>.EQ.H,32)         ; Jump to RX2 if byte received is a "2",
(JUMP,RX3,$POT<TMP>.EQ.H,33)         ; Jump to RX3 if byte received is a "3",
(JUMP,GETC)                          ; otherwise, continue to look for them.
(DENT,RX1)                           ; A "1" is received.
X10.                                 ; Move the X axis 10 inches in the positive
                                     ; direction.
(JUMP,GETC)                          ; Look for more characters.
(DENT,RX2)                           ; A "2" is received.
X-10.                                ; Move the X axis 10 inches in the negative
                                     ; direction.
(JUMP,GETC)                          ; Look for more characters.
(DENT,RX3)                           ; A "3" is received.
M2                                   ; End of The Program
```

PORT

NOTES:

Prior to using the **PORT** command, an appropriate amount of memory must have been allocated by the **MALC** <3n>command (Each n = 1 byte).

RELATED COMMANDS:

$POT, MALC

PUSH

NAME:

PUSH - Push Data Into User's Stack

FUNCTION:

The **PUSH** command is used to store data in the program stack.

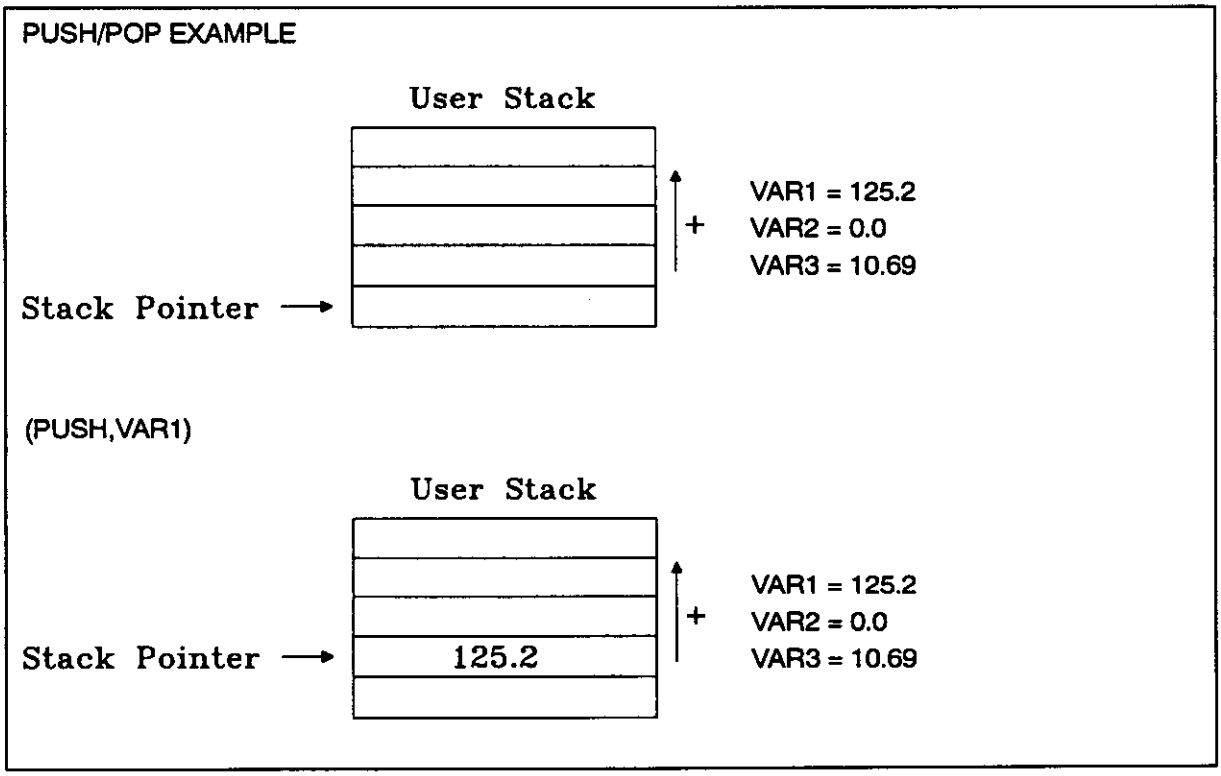
FORMAT:

(PUSH,data1,data2)

Data is stored in the data stack in a "last in - first out" basis.

RETURNS:

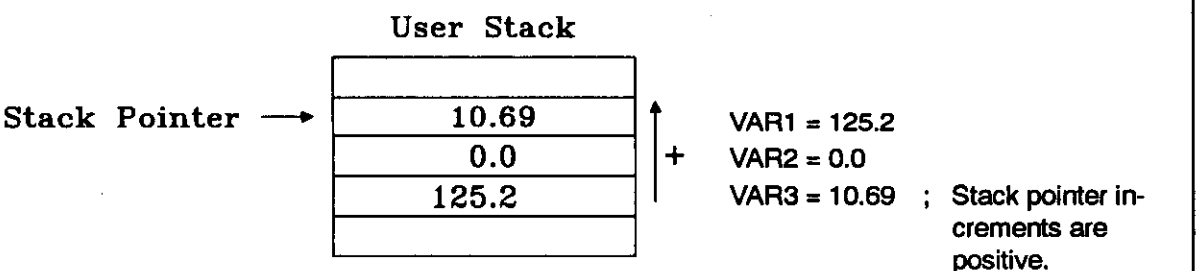
EXAMPLE:



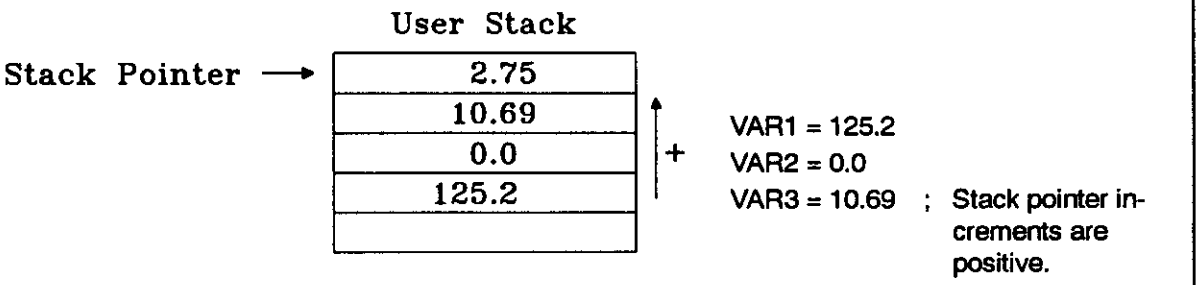
PUSH

EXAMPLE (CON'T):

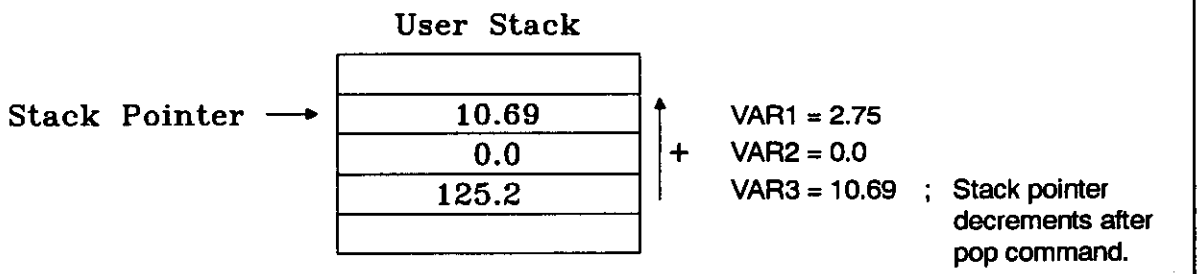
(PUSH,VAR2,VAR3)



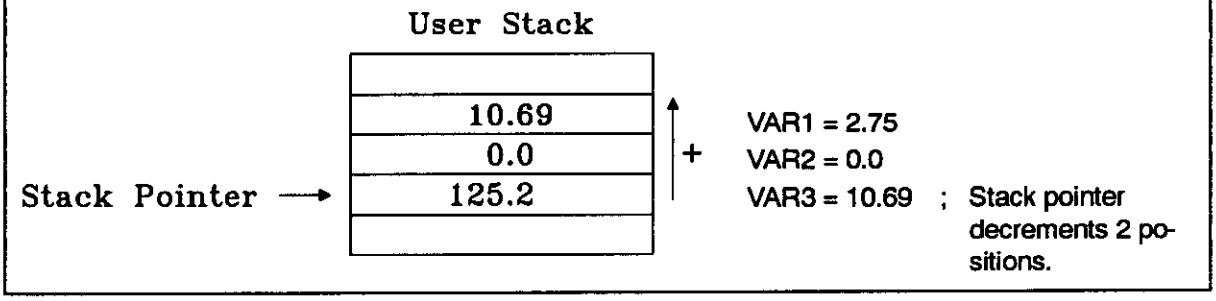
(PUSH,2.75)



(POP,VAR1)

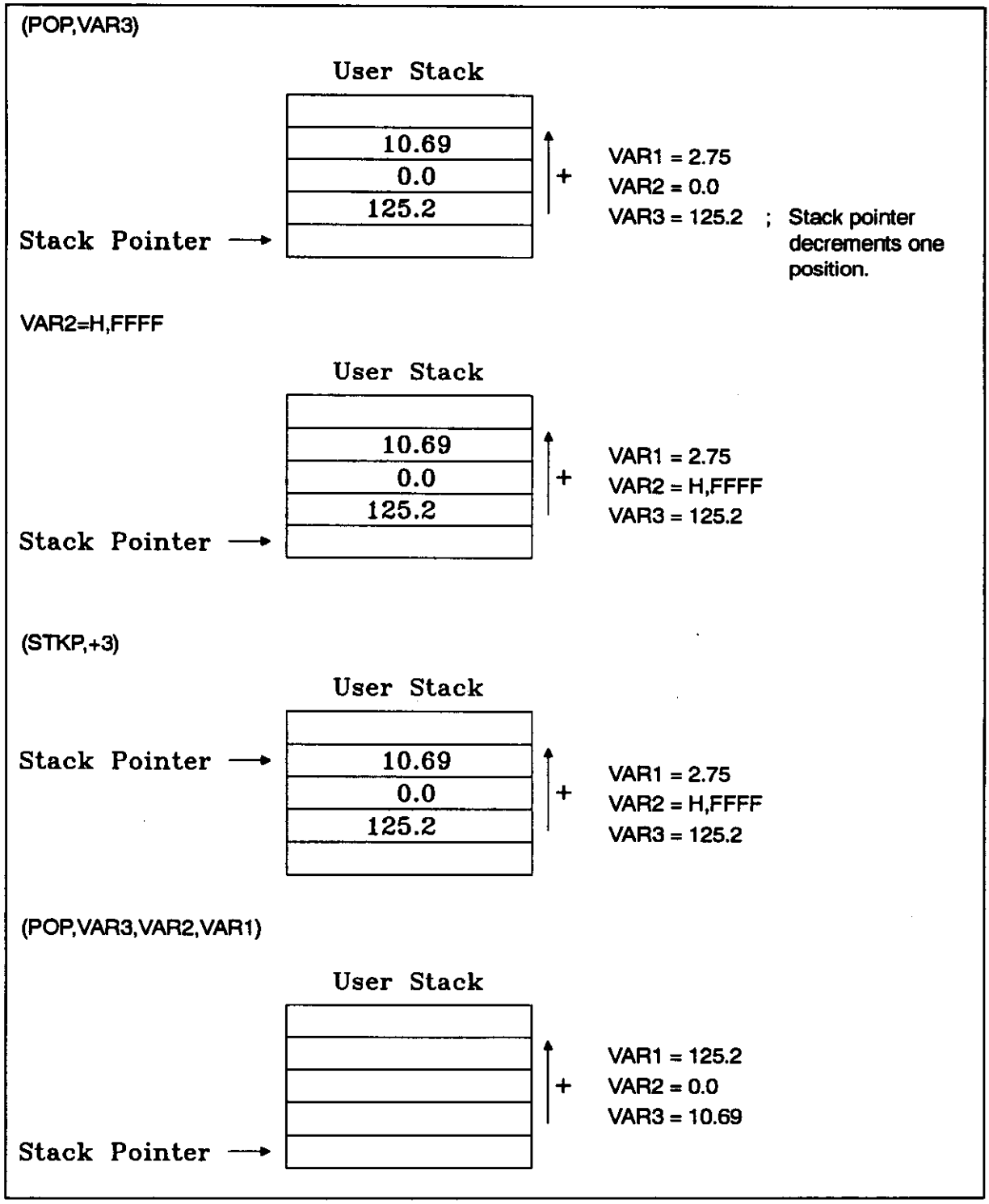


(STKP,-2)



PUSH

EXAMPLE (CON'T):



PUSH

NOTES:

Each data string stored by the **PUSH** command requires 4 bytes of User programmable memory. Program stack size (bytes) is established within the Parameter Mode. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

RELATED COMMANDS:

POP, STKP

RAMP

NAME:

RAMP - Define Axis Ramping time

FUNCTION:

The **RAMP**command is used to globally set the Ramping time of all axes, overriding the default Main System Parameter #30 setting established under the Individual Axis Parameter Mode (see the *Unidex 21 User's Manual* for a detailed description). This command is modal, and remains in effect until a subsequent **RAMP** command is issued, or the system is reset. Ramp time, as implemented in the Unidex 21 is the Acceleration/Deceleration time utilized during contouring or coordinated axes motions such as **G1**, **G2**, **G3**, or **G5**. The ramped trajectory may be either modified parabolic or linear and can also be affected by utilizing the Digital Filter Main System Parameter #60 (see also the *Unidex 21 User's Manual*) or through the **FILT** command. **G8** commands with feedrate charges utilize the Ramp time to accelerate or decelerate to the next velocity.

FORMAT:

(**RAMP**,ramping time)

Ramp values may be designated in two ways:

```
without a "."      ; The Ramping time value will be in milliseconds.
with a "."         ; The Ramping time value will be in seconds.
```

RETURNS:

EXAMPLES:

```
(RAMP,1000)        ; The Ramping time of all axes will be 1000 msec.
                     (no decimal point)

(RAMP,1.000)       ; The Ramping time of all axes will be 1.0 sec.
                     (decimal point used)

(DVAR,RMP)         ; Define the variable RMP.
RMP=2.5            ; Set the value of the variable RMP.
(RAMP,RMP)         ; Utilize the variable to set the Ramp time of all axes to 2.5
                     seconds.
```

RAMP

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #2 located in Appendix 3 of this manual.

NOTES:

1) Ramping time must be 1 to 32,767 msec.

2) The Ramp time is used for coordinated axis motion. Uncoordinated motion commands (**FXOF, G0, HOME, MORG**) use the **ACDE** rate instead.

RELATED COMMANDS:

ACDE, COEF, DVAR, FILT, FREE, G8, G9, G23, G24, TRAJ

RECO

NAME:

RECO - Record Axis Motion

FUNCTION:

The **RECO** command is used to initiate a Teach Mode during manual axes manipulation to sample axes positions. The positions are recorded in the User's memory. Typically, a User would disable the axes torque via the **MTOR** command, enable the recording function, and then move the axes. A good application for the usage of the **RECO** command is for recording the motions of a human machine operator, and then replaying the motion which has been recorded.

The hardware input signal from the triggering device is input at Joystick Port J2A. The signal at this port is monitored during each sample and may be configured to record only upon receipt of either a HIGH or LOW signal. The default is for recording to occur at a HIGH signal from Port J2A.

FORMAT:

(**RECO**<axis name,axis name,....>,rate,deadband,1/0)

or

(**RECO**,0)

axis name	The names of the axes whose position is to be recorded. Must be enclosed in < >.
rate	The frequency at which position sampling occurs. (1 to 255 msec)
deadband	The minimum number of Machine Steps of position change that is to be considered a position change.
1/0	Establishes the record signal from J2A as HIGH or LOW 1 - recording will occur only when J2A goes HIGH 0 - recording will occur only when J2A goes LOW

RETURNS:

RECO

EXAMPLE:

```
(RECO,<X,Y>,50,100,1)     ; When the signal of J2A goes HIGH, the positions of the X and
                            Y axis will be recorded every 50 msec. The deadband is 100
                            Machine Steps.

(RECO,0)                  ; Disable the position recording.
```

NOTES:

1) Prior to using the **RECO** command an appropriate amount of memory must have been allocated by the **MALC,<2,n>** command. (Each n = 256 bytes)

2) Position recording must be disabled before attempting to use the **PLAY** command.

3) To store the recorded memory to a file see the *Unidex 21 User's Manual* for a more complete description of the "File Mode".

RELATED COMMANDS:

MALC, MTOR, PLAY, REPT

REF

NAME:

REF - Send Axis to Hardware Home

FUNCTION:

The **REF** command is the same as the **HOME** command. Refer to the description of the **HOME** command contained in this manual.

FORMAT:

RETURNS:

EXAMPLE:

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Examples #5, and 6 located in Appendix 3 of this manual.

NOTES:

RELATED COMMANDS:

HOME

RETP

NAME:

RETP - Real Time Position Fetch

FUNCTION:

Sometimes it is necessary to record the axes position in synchronization with an external event. An example would be either flow detection in an ultrasonic test or pixel data acquisition from a camera system. The **RETP** command is used to initiate a Real Time Position Fetch. The Position Fetch event is triggered externally, by the User, through an I/O connector located on Unidex 21's rear panel (see Note 1). Data is latched in the Unidex 21's DSP Servo Control card with a latency of 3 to 8 microseconds for all 8 axes.

The position of the designated axis or axes is recorded in the User's memory (see Note 2), and may be transferred via RS-232 or disk for further analysis.

FORMAT:

(**RETP**,axis name,axis name,...)

RETURNS:

EXAMPLE:

```
(RETP,X,Y)          ; Enable a Real Time Position Fetch for the X and Y axes.

(RETP,0)            ; Disable a Real Time Position Fetch for all axes.
```

PROGRAMMING EXAMPLE:

The following program example shows Real Time Position Fetching with output to a data file.

```
(DVAR,ITER)         ; Define the local variable ITER.
(MALC,<2,100>)      ; Allocate memory for a Real Time Position function.
(RETP,X)            ; Enable a Real Time Position Fetch on the X axis.
G1 F100. X10.       ; Initiate a Linear Move.
(RETP,0)            ; Disable the Real Time Position Fetch.
```

RETP

PROGRAMMING EXAMPLE (CON'T):

```
(OPEN,W,TST.TMP)          ; Open the external file to write data to.
(MEND,W)                  ; Set the pointer to the top of the file.
ITER=1                    ; Set the iteration counter to 1.
(RPT,$RTP<0>              ; Repeat Loop
                          ; Note that the $RTP is a pointer to the Real Time Position
                          ; buffer. The first element in this buffer ($RTP<0>) indicates
                          ; the size of the buffer, subsequent buffer elements are the
                          ; stored axis position .If multiple axis positions are desired
                          ; i.e.,(RETP,X,Y) then the data is stored as follows for N
                          ; position fetches:
                          ;   $RTP<0> = Number of elements in buffer
                          ;   $RTP<1> = First position fetch X axis
                          ;   $RTP<2> = First position fetch Y axis
                          ;   $RTP<3> = Second position fetch X axis
                          ;   $RTP<4> = Second position fetch Yaxis
                          ;   || || || || || || ||
                          ;   $RTP<2N-1> = Nth position fetch X axis
                          ;   $RTP<2N> = Nth position fetch Y axis
(WRIT,#BTF<$RTP<ITER>>)   ; Write data out to the file. Note that the Position Fetch data
                          ; is stored internally in a Binary or Hex format.
ITER=ITER+1               ; Point to the next element in the buffer.
)                         ;
(END,S)                   ; Close the file and save the data.
```

NOTES:

1) All User supplied interrupt signals must be debounced. This input is directly transferred into Unidex 21's DSP Servo Control card. Refer to a description of the miscellaneous I/O connector (P23) located on the Unidex 21 Interface panel.

2) Prior to using the **RETP** command an appropriate amount of memory must have been allocated by the **MALC**,<2,n> command. (Each n = 256 bytes)

RETP

NOTES (CON'T):

3) Real Time Position Fetch events may occur at 1 msec intervals maximum, even though the latency of acquisition time is only 3 to 8 microseconds. This is due to the data transfer rate limitations between the CPU and DSP card.

RELATED COMMANDS:

$HSI, DVAR, END, HSIE, MALC, MEND, OPEN, WRIT

ROTA

NAME:

ROTA - Parts Rotation

FUNCTION:

The **ROTA** command is used to define the plane and degree of angle for Part Rotation. The Parts Rotation programming feature permits the User to utilize a one shape Subroutine or move sequence and change the orientation without reprogramming or changing coordinate reference frames.

FORMAT:

(**ROTA**,axis plane for rotation,angle)

The angle required for entry must be converted to degrees, or may be a User variable.

A positive angle entry will produce a CCW angle and is referenced from the axes plane. The entry is modal such that rotation will be in effect until deactivated.

An angle entry of zero will deactivate Parts Rotation.

RETURNS:

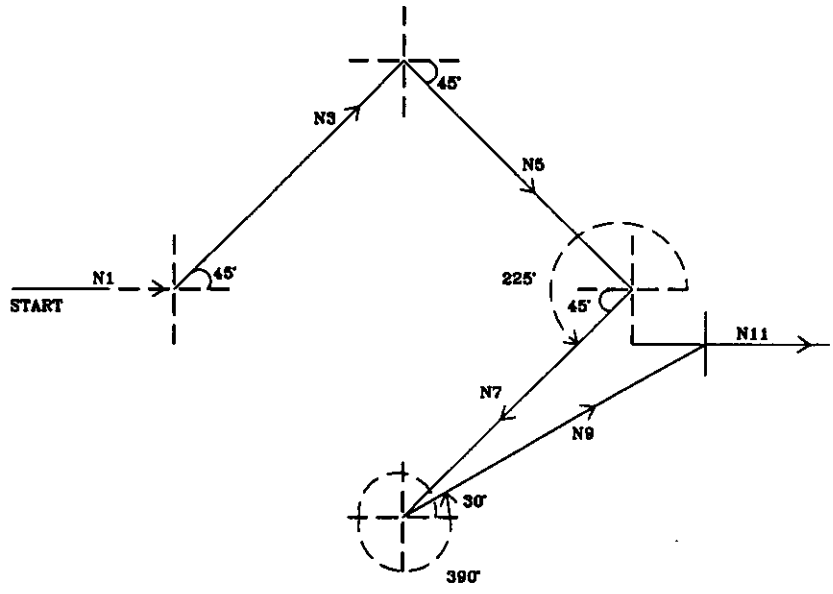
EXAMPLE:

(**ROTA**,X,Y,45)	; The Part Rotation angle will be set to 45° in the XY plane.
(**ROTA**,X,Y,VAR1)	; The Part Rotation angle will be set to the value of VAR1.
(**ROTA**,X,Y,0)	; Deactivate the Parts Rotation.

ROTA

EXAMPLE (CON'T):

LINE	COMMAND	DESCRIPTION
N1	G1 X5. F1000.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N2	(ROTA,X,Y,45)	Set the X,Y Part Rotation angle to 45°.
N3	X5.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N4	(ROTA,X,Y-45)	Set the X,Y Part Rotation angle to -45°.
N5	X5.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N6	(ROTA,X,Y,225)	Set the X,Y Part Rotation angle to 225°.
N7	X5.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N8	(ROTA,X,Y,390)	Set the Part Rotation angle to 390° (30°).
N9	X5.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N10	(ROTA,X,Y,0)	Disable the X,Y Part Rotation.
N11	X5.	Move the X axis five units (in/mm) in the positive (CCW) direction.
N12	M2	End of The Program



ROTA

PROGRAMMING EXAMPLE:

```
G70                                  ; Initiate English Programming (inches).
(DVAR,ANG1,ANG2,VAR1,VAR2,
 VAR3,VAR5,INCA,INCB,INCX,INCY)      ; Define the listed variables.
G1 X1. Y3. F200.                     ; Linear Move for the X and Y axes.
VAR2= 2. INCX= 0. INCY= 0.           ; Initialize variables, VAR2 is radius (2).
ANG1=180                             ;
(RPT,2                               ; Repeat the Loop once.
  INCA=INCX INCB=INCY                ; X and Y Offset
  ANG2=ANG1-(360./20.)               ; Angle 2 is the angle Offset.
  G91 G1 X=INCA Y=INCB F300.         ; Initiate Relative Positioning
                                     ; for  X and Y =  variables.
  (RPT,20                            ; Start the Repeat Loop.
    G1 X=INCX                        ; Initiate a Linear Move for X variables.
    (ROTA,X,Y,VAR5)                  ; The Part Rotation angle will be set to the
                                     ; value of VAR5.
    G1 G91 X.4                       ; Move the X axis in the positive direction.
    X-.2                             ; Move the X axis in the negative direction.
    Y-.2                             ; Move the Y axis in the negative direction.
    Y.4                              ; Move the Y axis in the positive direction.
    Y-.2                             ; Move the Y axis in the negative direction.
    X-.2                             ; Move the X axis in the negative direction.
    (ROTA,X,Y,0)                     ; Reset the Part Rotation angle.
    G2 L=VAR2 C=ANG1 D=ANG2          ; Position the angle such that:
                                     ;   L=Radius
                                     ;   C=Beg Angle
                                     ;   D=End Angle
    VAR1=360./20.                    ; Define the angle increment (18°).
    VAR5=VAR5-VAR1                   ; Establish a new angle for ROTA command.
    ANG2=ANG2-VAR1                   ; Increment the ending angle.
    ANG1=ANG1-VAR1                   ; Increment the beginning angle.
  )                                  ; Close the nested Repeat Loop.
```

ROTA

PROGRAMMING EXAMPLE (CON'T):

```
  VAR2=VAR2-.5          ; Decrease the radius of VAR2.
  INCX=0 INCY=0         ; Reinitialize the variables.
  G1 X-1. Y-3.          ; Linear Move of the X axis in the negative
                        ; direction, and the Y axis in the negative
                        ; direction.
  G1 X1.5  Y3.25        ; Linear Move of the X axis in the positive
                        ; direction, and Y axis in the positive direction.
  VAR5=VAR5-9           ; Decrement the variables.
  ANG1=ANG1-9           ;
)                       ; Close the outer Repeat Loop.
M0                      ; Program Stop
```

NOTES:

RELATED COMMANDS:

DVAR, MIR

RPT

NAME:

RPT - Repeat Loop

FUNCTION:

The **RPT** command is used to initiate a Repeat Loop. The repetition of the loop will be determined by the repeat count number via direct numeric data or a User variable. A Repeat Loop may also be nested within a Repeat Loop.

FORMAT:

(**RPT**,repeat count number or User variable
    data
    data
)

The **RPT** command and loop closure parentheses must be placed on separate lines, no other code can be inserted. The last line should contain the ending parentheses only.

RETURNS:

EXAMPLES:

```
(RPT,10                          ; The entire loop will be repeated 10 times.
  X1000.
)                                ; Close the Repeat Loop.
```

Repeat loops may also be nested together as follows:

```
(RPT,8                           ; The outer loop will be repeated 8 times.
  G1 X1000. Y1000. F100.
  (RPT,VAR1                      ; The inner (nested) loop will be repeated based upon the value
                                   of User variable VAR1.
    Z1500.
    X-1000. Y-1000.
  )                              ; Close the nested Repeat Loop.
)                                ; Close the outer Repeat Loop.
```

RPT

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #2 located in Appendix 3 of this manual.

NOTES:

The Repeat function requires 16 bytes of stack memory which is taken from the User RAM during program execution.

RELATED COMMANDS:

DVAR

RTRS

NAME:

RTRS - Retrace

FUNCTION:

The **RTRS** command provides the User with the ability to Retrace a program flow directly along the originally programmed path. The **RTRS** command must only be used with programs and/or program blocks that are repeatable in reverse. Refer to the notes on this command for specific programming considerations when using the Retrace function. This function may also be activated from the membrane panel retrace key or <cntrl> R from an AT style keyboard. When enabled in this manner, the Unidex 21 automatically forces the system into a single block mode of operation.

FORMAT:

(**RTRS**,enable/disable)

The Retrace function is enabled by placing a non-zero number next to the command.

The Retrace function is disabled by placing a zero next to the command.

RETURNS:

EXAMPLE:

```
N1 (RTRS,0)          ; The program will run in a normal forward motion, N2 - N5.
N2 program block
N3 program block
N4 program block
N5 program block
N6 (RTRS,1)          ; Retrace will begin, program blocks N5 - N2 will be performed.
                       When N1 is reached, program will proceed forward, N2 - N5.
                       Back and forth motion will result.
```

RTRS

EXAMPLE (CON'T):

```
N1  (DVAR,VARB)          ; Defines the variable VARB.
N2  (RTRS,0)             ; Ensure can't retrace past this point.
N3  program block        ; Program will run in a normal forward motion
N5  program block        ; for blocks N3 - N8.
N6  program block        ;
N7  program block        ;
N8  program block        ;
N9  VARB= BTF($IN0)      ; $IN0 will determine if the program will retrace.
N10 (RTRS,VARB)          ; If $IN0 was HIGH, N9 - N2 will be retraced
N11 M2                   ; else, End of The Program.
```

NOTES:

Most commands may be used with the Retrace function and produce the desired results. The following items provide programming constraints that must be acknowledged prior to using the Retrace function for certain special commands..

1) The following commands provide the same functions during Retrace as they do during a standard program run.

HOME	See Note 7
REF	
RTRS	
SLEW	

2) During a Retrace the following commands are adjusted to their complementary function:

CLS	Following the Subroutine the return point will be one block above CLS command.
DFLS	Library Subroutine flow will be in reverse.
DFS	Subroutine flow will be in reverse.

RTRS

NOTES (CON'T):

POP	The PUSH function will be performed.
PUSH	The POP function will be performed.
RPT	Repeat Loop flow will be in reverse.
STKP	The stack pointer index sign will be reversed.

3) When the **$R** system command is encountered during a Retrace, the stack pointer is internally returned to the upper data set before the command is read. To maintain the correct data/name relationship each **$R** command should occupy its own program block. For Example:

VAR1=$R VAR2=$R	During Retrace VAR1 and VAR2 data will be correctly assigned.
VAR1=$R VAR2=$R	During Retrace VAR1 and VAR2 data will be assigned incorrectly.

4) To perform ICRC commands during a Retrace, the appropriate Retrace condition must be provided within the command. For example:

G40	The previous mode was G40.
G41 X1.	During initial program run the G41 command establishes a left cutter Offset of X1. During Retrace the Unidex 21 will read G40 in the previous motion block to deactivate ICRC with an end move of X1.
G41 X1.Y1.	G41 has been previously established for Retrace.
G40 X2.	During initial program run the G40 command will deactivate ICRC with an end move of X2. During Retrace the Unidex 21 will read the G41 command in the previous motion block to establish a right cutter Offset of X2.
G42 X2.	During initial program run the G42 command establishes a right cutter Offset of X2. During Retrace the Unidex 21 will read G40 in the previous motion block to deactivate ICRC with an end move of X2.

RTRS

NOTES (CON'T):

G42 X1. Y2.	G42 has been previously established for Retrace.
G40 X2.	During initial program run the G40 command deactivates ICRC with an end move of X2. During Retrace the Unidex 21 will read the G42 command in the previous motion block and establish a left cutter Offset of X2. (NOTE: While in Retrace the Unidex 21 treats G42 commands as G41 commands.)

5) The following commands provide the same function during Retrace as they do in a standard program run. Appropriate Retrace motion, however, will occur only if the command's complement is contained in the preceding program block. For example:

(SCO,X0,Y0)	Appropriate Retrace motion is provided without
(SCO,X1,Y1)	affecting the standard program run.

ACDE	**MTOR**
AFCO	**PID**
CCP	**PLC**
COEF	**PLNE**
COMM	**RAMP**
CPAG	**ROTA**
DRUN	**SCF**
FREE	**SCO**
HWEL	**SIOC**
INT1/INT2	**TERM**
LINK	**TRAJ**
MIR	**TRAK**
MSG	**UMFO**
MSTD	**ZONE**

RTRS

NOTES (CON'T):

6) The following commands cannot be contained in a program or program segment for Retrace since they are valid in one direction only:

ABTS	**MALC**
CFIT	**MEND**
ELPS	**OPEN**
END	**PLAY**
HSIE	**RECO**
JOIN	**RETP**
JUMP	**WRIT**

7) The following commands cannot be retraced because of the reasons provided:

FXOF	The UCO4 (A4) flags are confused in the Absolute Mode.
G92	When Position Registers are established, the previous program position is erased. Any Incremental (G91) moves that follows will be performed correctly but any Absolute (G90) move will not.
HOME, REF	When a HOME move is requested, the previous program position is erased. Any Incremental (G91) move that follows will be performed correctly but any Absolute (G90) move will not.
MORG	The UCO4 (A4) flags are confused in the Absolute Mode.
SCRB	Character scribing is a special function that cannot be performed in reverse.

RELATED COMMANDS:

SCF

NAME:

SCF - Scaling Factor

FUNCTION:

The **SCF** command provides a User with the ability to Scale Up or Scale Down an individual axis. Errors will not accumulate from move to move when the Scaling Factor is being utilized. The **SCF** command does not effect an inch to metric conversion. The Unidex 21 executes the Scaling Factor prior to making the conversion. Once **SCF** is set it must be enabled through the use of **SCO** command.

FORMAT:

(**SCF**,axis name and scaling factor)

The Scaling Factor must be a positive number ranging in value from .00001 to 99.99999.

RETURNS:

EXAMPLE:

(**SCF**,X0.33333,Y1.5)	; The Scaling Factor of the X axis is 0.33333, and the Scaling Factor of the Y axis is 1.5.
(**SCF**,X=VAR1,Y=VAR2)	; The Scaling Factor will be set to the values of variables VAR1 and VAR2.

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #1 located in Appendix 3 of this manual.

NOTES:

1) The **SCF** command is modal.

2) The default Scaling Factor is 1.

SCF

NOTES (CON'T):

3) Circles cannot be scaled into the form of an ellipse through use of the **SCF** command. Only the individual points are affected. For example:

G2 X2. Y4. I6. J8.

becomes

G2 X1. Y1. I3. J2.

after

(**SCF**, X.5, Y.25)

4) **G4** Dwell commands are not affected by scaling.

RELATED COMMANDS:

DVAR, ELPS, SCO

SCO

NAME:

SCO - Scaling ON/OFF Control

FUNCTION:

The **SCO** command provides a User the ability to enable or disable the Scaling Factor. Scaling is operative in both the Incremental and Absolute mode. When in the Absolute Mode, keep in mind that this function is referenced to the "Home" as established by **G92**. Prior to enabling **SCO**, the individual axes Scale Factor must be enabled by the **SCF** command.

FORMAT:

(**SCO**,axis name and enable/disable scaling function)

The Scaling function is enabled by placing a non-zero number next to the axis name.

The Scaling function is disabled by placing a zero next to the axis name.

RETURNS:

EXAMPLE:

(SCO,1)	; Enable all Scaling Factors.
(SCO,0)	; Disable all Scaling Factors.
(SCO,X0,Y1)	; Disable the X Scaling Factor, and enable Y Scaling Factor.
(SCO,X=VARA,Y=VARB)	; Enable or disable scaling on the XY axes based upon the value of variables VARA and VARB.

PROGRAMMING EXAMPLE:

For an example of how this command may be used, refer to Programming Example #1 located in Appendix 3 of this manual.

SCO

NOTES:

1) When an interrupt is enabled within a program, depending on the interrupt type, the program may skip the remainder of the program block. If this occurs, remember the current Machine Position and go to a Subroutine or entry point. It is then necessary that the User reestablish the Software Home before Scaling is reenabled. Reestablishing the Software Home is necessary since when the Unidex 21 returns from the Subroutine or entry point, it remembers the current Machine Position and the Scaling is no longer referenced to the prior Software Home.

2) The **SCO** command is modal.

3) The default is a disabled Scaling function.

RELATED COMMANDS:

DVAR, SCF

SCRB

NAME:

SCRB - Character Scribing

FUNCTION:

The Unidex 21 provides a Character Scribing facility as a standard feature. Block letters, numbers, and some ASCII characters are included. The **SCRB** command initiates Character Scribing. Prior to using this feature, the CHARACTER.$$$ file must be loaded into the User's memory (see Note 1). The following names are used within the CHARACTER.$$$ file and must NOT be redefined anywhere in the Main Program:

SCRB< >	Array name
SCRB	Entry name
S$RB	Subroutine name

FORMAT:

(**SCRB**,option,variable/text)

where

option 0 – outputs text
1 – converts variable to (±) 8 digit integer with leading zero
2 – converts variable to (±) 8 digit integer without leading zero

Character Scribing is capable of twenty characters per call.
The character ")" is scribed by using \)\.

RETURNS:

EXAMPLE:

(SCRB,AEROTECH)	; Each character of AEROTECH will be scribed.
(SCRB,1,VAR)	; Variable is converted to a ± 8 digit integer with a leading zero.
(SCRB,2,VAR)	; Each variable is converted to a ± 8 digit integer with no leading zero.

SCRB

NOTES:

1) The CHARACTER.$$$ file may be loaded into the User's memory from the Edit Mode's "Get File" function. The program will always exist in the Unidex 21 firmware, but is not accessed unless requested in order to save User RAM. The CHARACTER.$$$ file may be edited in the same manner as a program.

2) When using options 1 and 2, the variable is always interpreted as a floating point number. This number will be rounded to the nearest integer value.

RELATED COMMANDS:

DVAR, SCF, SCO

SIOC

NAME:

SIOC - System I/O Control

FUNCTION:

The **SIOC** command controls the system I/O by indicating either to wait or not to wait before processing I/O code. The Unidex 21 typically operates in a block lookahead mode which means that a non-motion command may be processed before an in-process move command is completed.

The I/O functions that are under **SIOC** control are:

$INn
$OTn
$000 to **$7FF**
$800 to **$F6F**
$F70 to **$FAF**

FORMAT:

(**SIOC**,n)

If n equals zero, the program will wait until the last block of instructions are complete.

If n does not equal zero I/O code is processed immediately after it is decoded.

RETURNS:

EXAMPLES:

```
(SIOC,0)
G1 X10. Y10.          ; The command block will be completed prior to
$OT7= H,1               processing the I/O function.

(SIOC,VAR1)
G1 X10. Y10.          ; The I/O function $OT7=H,1 will be processed before the
$OT7=H,1                command block completion if the value of variable VAR1
                        is non-zero.
```

SIOC

NOTES:

The default status of **SIOC** is zero, the last block of the program will be completed prior to processing the I/O code.

RELATED COMMANDS:

$INn, $OTn, $000-$FAF, DVAR, INT1/INT2

SKEY

NAME:

SKEY - Define Custom Softkey

FUNCTION:

The Unidex 21 provides two methods in which to program the Function Keys F1 - F8 for custom definition. The methods are as follows:

method 1– utilizes the Function Keys to perform batch mode programming, system level key strobe emulation, or User interrupt 1/2 activation. The settings of these keys are established under the Main System Parameter #301. These System Level Function Keys can be extremely useful, and if desired, multiple levels may be initiated by loading different parameter files. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

method 2– is implemented utilizing the **SKEY**, **SKLV**, and **SKYD** commands. This method provides a more enhanced way of programming the F1 - F8 Function Keys within a User's Parts Program. Up to 100 levels of each Function Key may be utilized to create totally menu driven operation. The Software Programmable Function Keys can be interactive with the System Level Function Keys (method 1 above). Also, a program containing function key commands may be joined into a Main Program via the **JOIN** command. Of course, variables may also be utilized within the Softkey **SKEY** commands.

FORMAT:

The explanation will be divided into two parts for easier reference, a Detailed Explanation and a Summary.

SKEY

DETAILED EXPLANATION:

(**SKEY**,intn#,label,title,key#,level#,function#)

where

intn#/label

intn# 0 – Delete this Softkey. Note that neither the entry field nor the title field is necessary when using this number. However, if these fields are not specified, spaces must be included within the command as placeholders. (See Example)

1 – Enable Softkey. Unidex 21 will not stop or abort the current move when jumping to the entry point specified by label. It will perform defined functions (I/O, mathematical, system variable input/output, or **MST**). Note that there is a great flexibility in the usage of the entry field with this option. The entry point format specified determines the option being used. Refer to the following examples.

a) **M,S,T** output is to be performed

entry point format: **MFxxxx** or **SFxxxx** or **TFxxxx**

where:

M,S,T specifies the appropriate output bus

and

xxxx specifies the value to be output on that bus. (Hexadecimal number)

b) Level is to be changed only;

By simply omitting the entry point (i.e., label) all together, detection of a Softkey will initiate a Softkey level change only (refer to function below). Note that a space is required as a placeholder for the entry point.

SKEY

DETAILED EXPLANATION (CON'T):

2 – Enable Softkey as **INTN2** user interrupt option. The label refers to the Subroutine name. Unidex 21 will stop the current move and abort any functions remaining in this block. The current Machine Position will be stored. Upon completion of the specified Subroutine, the Unidex 21 will return to the next block of the program.

The Subroutine specified must be defined elsewhere in the program using either the **DFLS** or **DFS** commands.

3 – Enable Softkey as **INTN3** user interrupt option. The label refers to the entry point location. Unidex 21 will stop the current move and abort any functions remaining in this block. It will then jump to the specified entry point.

The entry point specified must be defined elsewhere in the program using the **DENT** command.

4 – Enable Softkey as **INTN4** user interrupt option. The label refers to the Subroutine name. Unidex 21 will finish all of the functions in the current block before going to the specified Subroutine. Upon completion of the routine, the Unidex 21 will return to the next block of the program.

The Subroutine specified must be defined elsewhere in the program using either the **DFLS** or **DFS** commands.

5 – Enable Softkey as **INTN5** user interrupt option. The label refers to the entry point location. Unidex 21 will finish all of the functions in the current block before jumping to the specified entry point.

The entry point specified must be defined elsewhere in the program using the **DENT** command.

title – This specifies a character string which will be displayed at the F1 - F8 Function Keys. The title can be 1 to 15 characters in length.

SKEY

DETAILED EXPLANATION (CON'T):

key – Key number within the specified level. This number must be less than or equal to the number of keys per level (defined with the **SKYD** command).

Note that if there are more than 8 softkeys per level, the keys are displayed in groups of seven. Softkey number 8 (F8) is used to move between the different groups. The Unidex 21 will display

F8 -- ETC --

In this case, the softkeys are referenced 1-7 (F1-F7), 8-13 (F1-F7), etc., (i.e., F8 is not accessible). However, if there are exactly eight softkeys per level, F8 is available for general purpose use.

level – This is the level for which the specified key is to be active. This number must be less than or equal to the number of levels defined with the **SKYD** command.

function – This feature permits the User to change levels and/or change groups within the current level. The possible options are as follows:

option 0 – Execute only. No level change

1 – Execute and disable this Softkey. The key will be re-enabled any time this level is entered from a higher level, (Level 0 is the highest). It cannot be re-enabled from a lower level.

2 – Execute and move to next key group if available (i.e., if -- ETC -- is displayed).

3 – Execute and increment level number to lower level.

4 – Execute and decrement level number to higher level.

5 – Execute and move to level found on Softkey stack. Stack pointer is adjusted accordingly.

SKEY

DETAILED EXPLANATION (CON'T):

6 – Execute and reinitialize the Softkey stack pointer.

7 – Execute and blank out all softkeys. All machine sub-modes are disabled (refer to **SKLV,N**).

8 – Execute and re-enable system Function Keys. All machine sub-modes are re-enabled (Refer to **SKLV,S**).

9 – Reserved

10-109 – Execute and move to level 0-99 (i.e., function-10)

Expanded Functions

The expanded functions listed below have been included to give the User some control over the Softkey stack. Enhance the function # below with 3, 4, and 10 to 109 cases such that:

256 + function# - Current level number is placed onto the Softkey stack. The stack pointer is adjusted appropriately.

512 + function# - Current level number is placed onto the Softkey stack. The stack pointer is not adjusted.

768 + function# - One Softkey stack entry is removed from the Softkey stack. The stack pointer is adjusted appropriately.

1024+ function# - The Softkey stack is cleared, and the stack pointer is reinitialized.

For Example:

259 - Execute and increment level number to lower level. The previous level is stored on the Softkey stack. The stack pointer is adjusted.

SKEY

DETAILED EXPLANATION (CON'T):

516 - Execute and decrement level number to higher level. The previous level is stored on the Softkey stack. The stack pointer is not adjusted.

778 - Execute and move to level 0. Discard 1 Softkey stack entry. The stack pointer is adjusted.

1044 - Execute and move to level 10. The Softkey stack will be cleared, and the stack pointer reset.

To make this feature even more useful, the function number may be expressed as a variable. The value of this variable will be evaluated when the key is pressed. To utilize this feature, the variable name must be preceded by a #.

SUMMARY:

(SKEY, intn#,label,title,key#,level#,function#)

intn# 0 – Delete this key.

1 – For **MST** output, entry =MF/SF/TFxxx or level change only, entry =,, or same as INTN interrupt option, entry = subroutine 2/3/4/5 same as INTN interrupt option.

label – Subroutine label (INTN# 2/4)
or entry point (INTN# 3/5)
or don't care (INTN# 0/1 or , ,)
or (INTN# 1)

where

MFxxxx is the output M function Mxxxx
SFxxxx is the output S function Sxxxx
TFxxxx is the output T function Txxxx

SKEY

SUMMARY (CON'T):

title – 1 to 15 characters for F1 - F8.

key# – 1 to 100, if > 8, display in groups of 7 with F8 - ETC-.

level# – 0 to 99, 0 is highest level.

function – This feature permits the User to change levels and/or change groups within the current level. The possible options are as follows:

option 0 – Execute only. No level change

1 – Execute, then clear this key, re-enable when higher level is activated.

2 – Execute, then change to ETC-, if more than 8 keys.

3 – Execute, then move to lower level.

4 – Execute, then move to upper level.

5 – Execute, then move to level in stack.

6 – Execute, then do (SKLV,I).

7 – Execute, then do (SKLV,N).

8 – Execute, then do (SKLV,S).

9 – Reserved

10 to 109 – Move to level 0 to 99.

SKEY

SUMMARY (CON'T):

Expanded Functions

The expanded functions listed below have been included to give the User some control over the Softkey stack. Enhance the function # below with 3, 4, and 10 to 109 cases such that:

256*1 + function# - Save the level to the stack, adjust the pointer.

256*2 + function# - Save the level to the stack, do NOT adjust the pointer.

256*3 + function# - Discard 1 stack, adjust the pointer.

256 * 4 + function# - Clear the stack, initialize the pointer.

```
(SKLV,nn)    ; Change to level nn, 0 - 99.
(SKLV,N)     ; Black out Softkeys and machine system Function Keys, do
               not permit jog/mdi/auto/single/slew/run/feedhold/quit/re-
               trace/abort/handwheel. Only permit cycle start/tracking dis-
               play/error acknowledge.
(SKLV,S)     ; Re-enable machine sub-mode and system Function Keys.
```

EXAMPLE:

This Softkey example illustrates the use of the Unidex 21 Softkey option. It uses three levels of softkeys with 14 keys per level (F1-F7, F1-F7). The User moves between the levels by pressing the desired Softkey.

```
(SKYD,14,3,128)  ; Define Softkey table to permit 3 Softkey levels with 14
                   softkeys per level. 128 bytes are allocated for the Softkey
                   stack.
(SKLV,I)         ; Ensure that the Softkey stack is initialized.
(SKLV,0)         ; Ensure that the current Softkey level is 0.
```

SKEY

EXAMPLE (CON'T):

```
LEVEL #0 GROUP #1
(SKEY,1, ,Laser Control,1,0,268)        ; F1 - Push the current Softkey level on
                                        ;      stack. Change to Softkey level #2.
                                        ;      Do NOT abort the current move.
(SKEY,1, ,Motor Control,2,0,3)          ; F2 - Increment to Softkey level #1.
                                        ;      Do NOT abort the current move.
                                        ; F3 - Not used
(SKEY,3,ENDP,Exit Program,4,0,7)        ; F4 - Jump to ENDP immediately. Don't wait
                                        ;      for the current block. Keep the Softkey
                                        ;      level the same. But delete all softkeys.
                                        ; F5 - Not used
                                        ; F6 - Not used
                                        ; F7 - Not used
                                        ; F8 - --ECT--

LEVEL #0 GROUP #2                       ; This level is not used


LEVEL #1 GROUP #1
(SKEY,4,GOX,Free Run X,1,1,0)           ; F1 - Call the Subroutine GOX after the
                                        ;      current block completes.
                                        ;      Do NOT change the Softkey level.
(SKEY,2,STPX,Stop Free Run X,2,1,0)     ; F2 - Call the Subroutine STPX now.
                                        ;      Do NOT wail til the block finishes.
                                        ;      Do NOT change the Softkey level.
                                        ; F3 - Not used
(SKEY,1, ,Exit Motor Control,4,1,4)     ; F4 - Exit laser control.
                                        ;      Change the Softkey level to that next
                                        ;      higher level (0). The current block is
                                        ;      not aborted
```

SKEY

EXAMPLE (CON'T):

```
LEVEL #1 GROUP #1 (CON'T)
(SKEY,1, ,Laser Control,5,1,259)        ; F5 - Push the current Softkey level on
                                        ;      stack. Increment to Softkey level
                                        ;      #2. Do NOT abort the current move.
                                        ; F6 - Not used
                                        ; F7 - Not used
                                        ; F8 - --ECCT--

LEVEL #1 GROUP #2
(SKEY,4,GOY,Free Run Y,8,1,0)           ; F1 - Call the Subroutine GOY after the
                                        ;      current block is completed.
                                        ;      Do NOT change the Softkey level.
(SKEY,2,STPY,Stop Free Run Y,9,1,0)     ; F2 - Call the Subroutine STPY now.
                                        ;      Do NOT wait till the block finishes.
                                        ;      Do NOT change the Softkey level.
                                        ; F3 - Not used
(SKEY,1, ,Exit Motor Control,11,1,4)    ; F4 - Exit the Laser Control. Change the
                                        ;      Softkey level to that next higher
                                        ;      level (0). The current block is not
                                        ;      aborted.
(SKEY,1, ,Laser Control,12,1,259)       ; F5 - Push the current Softkey level on stack.
                                        ;      Increment to Softkey level #2.
                                        ;      Do NOT abort the current move.
                                        ; F6 - Not used
                                        ; F7 - Not used
                                        ; F8 - --ECT--
LEVEL #2 GROUP #1
```

Note that this level may become active from either level 0 or from level 1. The previous level is found on the Softkey stack.

SKEY

EXAMPLE (CON'T):

```
LEVEL #2 GROUP #1 (CON'T)
(SKEY,1,MF80,Laser On,1,2,2)        ; F1 - Place H,80 on M-bus
                                    ;      then move to next Softkey
                                    ;      group. The level will not change.
                                    ;      Do NOT abort the current move.
                                    ; F2 - Not used
                                    ; F3 - Not used
(SKEY,1, ,Exit Laser Control,4,2,5) ; F4 - Exit the Laser Control. Change the
                                    ;      Softkey level to that when pressed
                                    ;      Laser Control. The current block is
                                    ;      not aborted.
                                    ; F5 - Not used
                                    ; F6 - Not used
                                    ; F7 - Not used
                                    ; F8 - --ECT--

LEVEL #2 GROUP #2
                                    ; F1 - Not used
(SKEY,1,MF90,Laser Off,9,2,2)       ; F2 - Placed H,90 on M-bus then move
                                    ;      to the previous Softkey group.
                                    ;      Do NOT change the level.
                                    ;      Do NOT abort the current move.
                                    ; F3 - Not used
(SKEY,1, ,Exit Laser Control,11,2,5) ; F4 - Exit the Laser Control and change
                                    ;      the Softkey level to that when
                                    ;      pressed Laser Control. The current
                                    ;      block is not aborted.
                                    ; F5 - Not used
                                    ; F6 - Not used
                                    ; F7 - Not used
                                    ; F8 - --ECT--
```

SKEY

EXAMPLE (CON'T):

BODY OF PARTS PROGRAM

```
(DVAR,VAR1)                          ; The Parts Program may contain any
(DENT,LOOP)                          ; commands necessary.
(MSG,<VAR1>,Enter 9 to exit now)     ;
G4  F.1                              ;
(JUMP,ENDP,VAR1.NE.9)                ;
(JUMP,LOOP)                          ;
(DENT,ENDP)                          ; Label for Program End
(FREE,X0,Y0)                         ; Ensure that both axes are OFF.
M90                                  ; Ensure that the laser is OFF.
(SKLV,S)                             ; Re-enable the machine sub-modes.
M2                                   ; End of The Program
                                     ;
(DFS,GOX                             ; Subroutine to start the X Free Run.
  (SKEY,0, , , 1,1,0)                ; Disable the Free Run Function Key "X".
  (FREE,X1,F10.)                     ; Start the X axis in continuous Free Run.
)                                    ;
(DFS,STPX                            ; Subroutine to stop the X Free Run.
  (FREE,X0)                          ; Stop free running on the X axis.
  (SKEY,4,GOX,Free Run X,1,1,0)      ; Reactivate the Free Run Function Key
                                     ; "X" again.
)                                    ;
(DFS,GOY                             ; Subroutine to start the Y Free Run.
  (SKEY,0, , ,8,1,0)                 ; Disable the Free Run Function Key "Y".
  (FREE,Y1,F10.)                     ; Start the Y axis in continuous Free Run.
)                                    ;
(DFS,STPY                            ; Subroutine to stop the Y Free Run.
  (FREE,Y0)                          ; Stop free running on the Y axis.
  (SKEY,4,GOY,Free Run Y,8,1,0)      ; Reactivate the Free Run Function Key
                                     ; "Y" again.
)                                    ; Label for Program End
```

SKEY

NOTES:

1) Prior to using the **SKEY** command a Softkey dimension must be defined utilizing the **SKYD** command.

2) The intn#, key#, level#, and function# can all be either BCD or variables where the value will be assigned when the **SKEY** command is decoded. A special case exists if the function # is programmed as "#VAR", which will assign the value when the key is actually pressed.

RELATED COMMANDS:

CPAG, DVAR, INT1/INT2, JOIN, SKLV, SKYD

SKLV

NAME:

SKLV - Change Custom Softkey level

FUNCTION:

This command permits the User to dynamically change the Softkey level, return the Function Keys to the system level, as well as clear the Softkey stack.

FORMAT:

(**SKLV**,option)

where

option

0-99 – Change to level option 0-99.

N – Disable all softkeys and machine mode sub-modes.

NOTE: No; jog/mdi/auto-single/slew/run/feedhold/home/quit/retrace/abort/handwheel is permitted. Only cycle-start/track/error ack are permitted.

S – Re-enable the machine mode sub-modes and system Function Keys.

I – Clear the Softkey stack, initialize the stack pointer.

RETURNS:

EXAMPLES:

```
(SKLV,0)          ; Set Softkey level to 0. (highest)
(SKLV,10)         ; Set Softkey level to 10. (assuming level defined)
(SKLV,N)          ; Blank out all softkeys and disable machine sub-modes.
```

SKLV

EXAMPLES (CON'T):

```
(SKLV,S)          ; Re-enable the softkeys and machine sub-modes.

(SKLV,I)          ; Clear the Softkey stack and reinitialize the stack pointer.
```

NOTES:

Refer to the **SKEY** command for a more comprehensive example, and further explanation.

RELATED COMMANDS:

DVAR, SKEY, SKYD

SKYD

NAME:

SKYD - Define Softkey table

FUNCTION:

The **SKYD** command, in conjunction with the related commands, permits the User to modify the actions taken when a Function Key is pressed from within a User Parts Program. The **SKYD** command allocates the memory needed to hold the Sotfkey table, as well as the Softkey stack.

FORMAT:

(**SKYD**,K,L,S)

where

K - specifies the number of softkeys per level (range: 1 - 100)

L - specifies the number of Softkey levels (range: 1 - 100)

S - specifies the number of bytes to reserve for the Softkey stack.

EXAMPLE:

```
(SKYD,10,10,100)          ; Defines the Softkey table with 10 keys per level and 10 levels.
```

In this example, 100 bytes were reserved for the Softkey stack.

NOTES:

1) The amount of User memory allocated is (K * L * 12) + bytes (i.e., each table entry requires 12 bytes).

2) The levels defined by this command are referenced 0 - (L - 1).

3) Refer to the **SKEY** command for a more comprehensive example, and further explanation.

RELATED COMMANDS:

DVAR, SKEY, SKLV

SLEW

NAME:

SLEW - Joystick/Teach Pendant/Trackball/Mouse Slew

FUNCTION:

The **SLEW** command enables the Joystick, Teach Pendant, Trackball, or Mouse to be called from within either a User's Parts Program or MDI Mode. Alternatively, in the Machine Mode the User can manually select the **SLEW** function. Or, when running a Parts Program in Single Mode, the User may select **SLEW** (or any other Machine Mode sub function) and then continue with the Parts Program execution.

In many instances, a User needs to perform an initial part alignment or acquire data points within a User Parts Program. The **$nAP** and **$nRP** system variables can be utilized to set the value of User defined variables for Part Program usage. This is different from digitizing, where a User may create a Parts Program by moving the axes to different points and recording the positional data. Please refer to digitizing in the "Edit Mode" section of the *Unidex 21 User's Manual.*

FORMAT:

(**SLEW**,Joystick/Trackball,active axes name)

RETURNS:

The Unidex 21 remains in the **SLEW** mode until the User quits by pressing either the button located on the Joystick handle or the middle button on a Trackball.

EXAMPLE:

```
(SLEW,X,Y,VERT,U)
      or
(SLEW,J,X,Y,Z,U)          ; Program run stops and axes X, Y, VERT, and U are active to
                            the Joystick control.

(SLEW,T,X,Y)              ; Program run stops and axes X and Y are active to the
                            Trackball/Mouse control.
```

SLEW

NOTES:

1) If neither Joystick/Teach Pendant nor Trackball/Mouse is specified, the axes will be active to Joystick/Teach Pendant control. Both Joystick/Teach Pendant and Trackball/Mouse cannot be active at the same time. The handwheels may be utilized in conjunction with either Joystick/Teach Pendant or Trackball/Mouse.

2) An axis that is not included in the **SLEW** command is disabled from Joystick or Trackball control.

3) The applicable Joystick/Trackball Main System Parameters are #26 and 27. (See the *Unidex 21 User's Manual* for further explanation.)

4) The applicable Joystick/Trackball Individual Axis Parameters are #18, 21, 22, and 23 established under the 401-408 grouping. (See also the *Unidex 21 User's Manual.*)

RELATED COMMANDS:

$nAP, $nRP, AFCO, HWEL

SYNC

NAME:

SYNC - Synchronize Two Unidex 21's

FUNCTION:

The **SYNC** command is used to permit handshaking between two Unidex 21 controllers. Utilizing this command will provide beyond 8 axes of control as well as control program flow of a Master/Slave arrangement. The communications link used is the RS-232 Terminal Port. Therefore, both systems must have front panels. Typically, a User might also tie together Port B of the Master with Port A of the Slave such that remote communication control and file transfer facilities would also be available.

FORMAT:

(**SYNC**,option,timeout)

where

option 0 – Cancels Synchronization.

1 – Sends Synchronized code to the Slave.

2 – Waits for Synchronize code from the Master.

3 – Permits automatic Synchronization after each program block. That is, the Master controller will send **SYNC** code after executing each program block, and the Slave waits for a Synchronized code before executing another block. Both must be operating in this mode.

and

timeout – Specifies the number of seconds the Unidex 21 should wait for the Synchronization code from the remote system. If timeout value is equal to zero, the timeout feature is disabled (i.e., waits forever).

SYNC

EXAMPLE:

```
(SYNC,1,0)      ; Send SYNC code to the remote system.
(SYNC,2,10)     ; Wait for a SYNC code from the remote system. If
                ; one is not received within ten seconds, notify the
                ; User of the error.
(SYNC,2,0)      ; Wait indefinitely for a SYNC code from the remote
                ; system.
(SYNC,3,5)      ; Synchronize with remote system after each program
                ; block. If feedback is not received within 5 seconds
                ; of completing the current block, notify the User of
                ; the error.
(SYNC,0,0)      ; Cancel the Synchronization.
```

NOTES:

1) The two units being Synchronized must have the **SYNC** code Main System Parameter set to a unique value (0/1). (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

2) When using options 0 and 1, the timeout period must be specified, but has no effect on command operation.

3) The wait for the **SYNC** code command, option 2, ignores all characters received previously. Therefore, if the Master sends the **SYNC** code before the Slave begins waiting for it, Synchronization will be lost.

RELATED COMMANDS:

STKP

NAME:

STKP - User's Stack Pointer Adjust

FUNCTION:

The **STKP** command provides the User with the ability to change the pointer location within the program stack.

When data is stored within the program stack by the **PUSH** command, the pointer is always located at the last stack entry. If a **POP** command is initiated, the data at the pointer is retrieved.

If the User is knowledgeable as to the location of the data in the stack, the **STKP** command in conjunction with the **POP** command may be used to retrieve data that is located at some point within the stack.

FORMAT:

(**STKP**,+/-nn)

The value selected for nn will be internally multiplied by 4.

RETURNS:

New program stack pointer.

EXAMPLE:

```
(STKP,+2)        ; Move the pointer down the stack 2*4 bytes.

(STKP,-3)        ; Move the pointer up the stack 3*4 bytes.
```

A more complete example is found with the **PUSH** command instruction.

NOTES:

RELATED COMMANDS:

POP, PUSH

TERM

NAME:

TERM - Terminal Display for MSG Function

FUNCTION:

The **TERM** command is used in conjunction with the **MSG** command to utilize the lower portion of the machine mode's display. The **MSG** command normally displays four lines only.

Enabling the **TERM** command disables the Tracking Display. The User may then use the lower part of the display below that which is used for the machine mode's function display.

FORMAT:

(TERM,nn)

If nn is zero the **MSG** display is limited to four lines only.

If nn is a non-zero number **MSG** may use the lower portion of the machine mode display.

RETURNS:

EXAMPLE:

```
(TERM,0)        ; The MSG display is limited to four lines.

(TERM,1)        ; The MSG may use the entire display.
```

NOTES:

The default setting is **(TERM,0)**.

RELATED COMMANDS:

CPAG, MSG

TRAJ

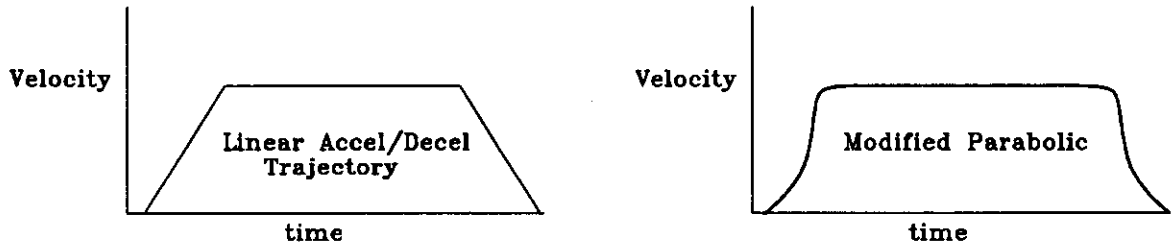
NAME:

TRAJ - Axis Acceleration/Deceleration Trajectory, Type Selection

FUNCTION:

The **TRAJ** command permits the selection of either Linear Trajectory or Modified Parabolic Trajectory, overriding the Axis Parameter Mode setting. The parameter values are not changed by this command. The values established by this command will be in effect until a subsequent **TRAJ** command, or the system is reset, at which time the parameter values are re-instated.

Normally, the User will want to have all axes set to the same trajectory type to provide fully coordinated motion. The Modified Parabolic type of Acceleration/Deceleration is often referred to as "anti-jerk". The Parabolic Trajectory Coefficient may also be changed through the **COEF** command which changes the linear velocity/time slope of the Parabolic Acceleration/Deceleration.



FORMAT:

(**TRAJ**,axis name and linear/parabolic,axis name and linear/parabolic....)

The trajectory will be Linear if a zero is placed next to the axis name.

The trajectory will be Parabolic if a non-zero number is placed next to the axis name.

RETURNS:

TRAJ

EXAMPLE:

```
(TRAJ,X0,Y1,Z=VAR1,U=$INn)    ; X axis has Linear Trajectory, Y axis has a Parabolic Trajectory, Z axis trajectory is dependent on value of VAR1, and U axis trajectory is dependent on Input bit "n" (1 or 0).

(TRAJ,0)                      ; All axes have Linear Trajectory.

(TRAJ,1)                      ; All axes have Parabolic Trajectory.
```

NOTES:

1) This command only affects coordinated motion commands such as **G1, G2, G3, G5, G8,** and **G9**, uncoordinated axis motion commands always use Parabolic Trajectory.

2) The applicable Main System Parameters are #57 and 59. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

   The applicable Individual Axis Parameter is #38 established under the 401-408 grouping. (See also the *Unidex 21 User's Manual* for further explanation.)

3) Usage of the **FILT** command or values in parameters #60 (exponential filter) may influence the actual trajectory under **G23** mode.

RELATED COMMANDS:

COEF, DVAR, FILT

TRAK

NAME:

TRAK - Position Tracking Display Control

FUNCTION:

The **TRAK** command permits the User to enable or disable the Position Tracking Displays for all the axes. This will speed up the system Parts Program execution by eliminating the update of the Position Tracking Displays which occurs approximately every 200ms. Under normal circumstances, it is not necessary to utilize **TRAK** for this purpose.

The **TRAK** command is also useful when the User wishes to display messages or gather input within a Parts Program, but does not want to have the operation confused by the scrolling of the Parts Program blocks.

FORMAT:

(**TRAK**,n)

If n= 0 – the Auto-Tracking Display will be OFF.

n= 1 – the Auto-Tracking Display will be ON.

n= 2 – the Auto-Tracking Display will be ON, however the Parts Program will not be displayed as it is run.

RETURNS:

EXAMPLE:

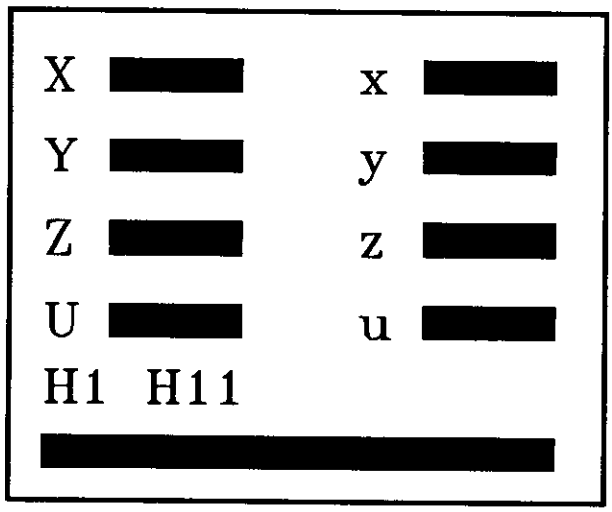
```
(TRAK,1)          ; The Auto-Tracking Display will be ON as the program is run.

(TRAK,VAR1)       ; Tracking Display option is set as per value of VAR1.
```

TRAK

NOTES:

Disabling the Tracking Display can free up the CPU for time which may be needed for processing motion commands.



NOTE: Shaded areas affected by TRAK command.

RELATED COMMANDS:

DVAR, MSG

UMFO

NAME:

UMFO - User Defined MFO Setting

The **UMFO** or Manual Feedrate Override command allows the User to vary the programmed feedrate up or down based upon a percentage, where 100% equals the programmed feedrate. Sometimes during a critical production process it is desired to prevent an operation from inadvertently changing the feedrate.

FUNCTION:

The **UMFO** command permits the User to set the percentage of Feedrate Override, and at the same time disable the Unidex 21's Front Panel **MFO** (Manual Feedrate Override) key. If the Front Panel **MFO** key is disabled, the desired percentage of the maximum Feedrate (0 - 200) must be included with the **UMFO** command.

FORMAT:

(**UMFO**,enable/disable,feedrate)

The **MFO** key is enabled by placing a zero next to the command.

The **MFO** key is disabled by placing a non-zero number next to the command.

RETURNS:

EXAMPLE:

```
(UMFO,1,100)                ; The MFO key is disabled and the Feedrate is set at 100%.
VAR1=75.5

(UMFO,1,VAR1)               ; The MFO key is disabled and the Feedrate is established in
                              VAR1 (75.5%).

(UMFO,0)                    ; The MFO key is enabled.
```

UMFO

NOTES:

1) The value of **UMFO** is set as a percentage value between 0 and 199 as a direct numeric entry. For example:

   75.5 equals 75.5%

   while the **$MFO** System Variable expresses this differently as

   0.755 equals 75.5%.

2) **MFO** affects all axes at the same time, and may be set to not affect the Free Run axes. The applicable Individual Axis Parameter is #36 established under the 401-408 grouping. (See the *Unidex 21 User's Manual* for a detailed description of the Parameter Mode.)

3) **MFO** may also be set up to affect the handwheel scaling. The applicable Main System Parameter is #55 (see also the *Unidex 21 User's Manual* for further detail).

RELATED COMMANDS:

$MFO, FREE, HWEL

ZONE

NAME:

ZONE - Safe Zone

FUNCTION:

The **ZONE** command is used to establish areas that an axis (axes) is not to travel. Safe Zones are used to prevent the tool from coming into contact with fixtures and to ensure that limits beyond the workspace are not exceeded.

The number of Safe Zones to be included in a program must be fixed at the beginning of the program with the **DZON** command. Zone parameters are referenced from the Software Home.

FORMAT:

(**ZONE**,zone name,enable/disable,zone parameters)

A Zone is enabled by placing a non-zero number next to the zone name.

A Zone is disabled by placing a zero next to the zone name.

RETURNS:

EXAMPLE:

```
(DZON,3)                                        ; Enables 4 sets of Safe Zones.

(ZONE,0,1,X-10.,X10.,Y-3.,Y3.)                  ; Zone 0 is enabled and zone parameters are
                                                  established.

(ZONE,2,0)                                      ; Zone 2 is disabled.

(ZONE,3,VAR1,X=VARA,X=VARB,Y=ARY<2>,Y=ARY<3>)   ; Zone 3 is enabled and
                                                  zone parameters are es-
                                                  tablished by variables and
                                                  arrays.
```

ZONE

NOTES:

1) The Safe Zones are only active under **G** code commands.

2) Safe Zone limits are not in effect when motion is initiated with the Home, Free Run, and Jog/Slew commands. The User may want to utilize **LIMT** instead.

3) The **ZONE** command is modal and remains until the system is reset or powered down.

RELATED COMMANDS:

DZON

CHAPTER 4: UNIDEX 21 MATH PACKAGE

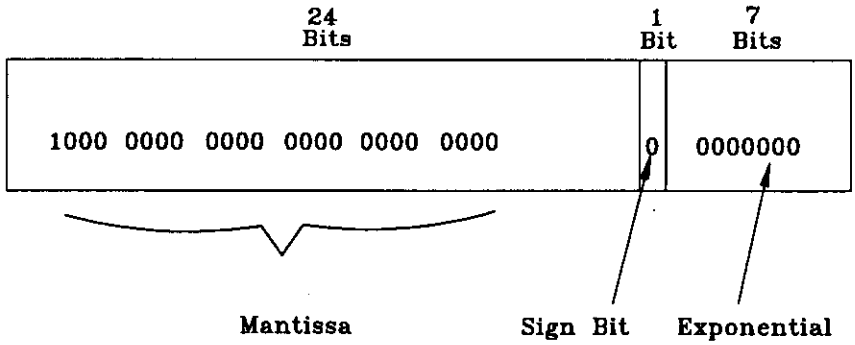
The Unidex 21 contains a powerful integral math package providing the User with full algebraic and logic capabilities which include addition, subtraction, multiplication, division, square root, trigonometric functions, and a variety of conversions.

SECTION 4-1: FLOATING POINT FORMAT

The Floating Point Format supports zero and non-zero values within the following range:

$9.22337177 \times 10^{18}$ >positive number >$5.42101070 \times 10^{-20}$

$-9.22337177 \times 10^{18}$ >negative number >$-2.71050535 \times 10^{-20}$



Mantissa — The Mantissa consists of 24 bits normalized (Most Significant Bit is always 1) Binary data.

Sign Bit — The Sign Bit designates the positive (0) or negative (1) value.

Exponential — The Exponent consists of 7 Bits of Binary data.

41H exponent = 1
40H exponent = 0
3FH exponent = -1
41H to 7FH are positive exponents
01H to 3FH are negative exponents

SECTION 4-2: BINARY NUMBER FORMAT

The Unidex 21 processes Binary Numbers in the following formats:

Hex Numbers

H,dddddddd ; d = 0 to 9
A to F

Character Numbers

"ABCD" ; Each character is processed through ASCII code

Hex numbers may contain 8 digits maximum.

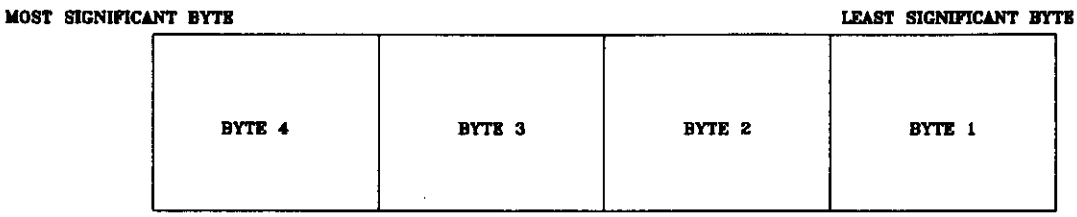
Refer to Appendix 4 (HEX NUMBERS AND EQUIVALENTS) for a decimal to Binary to Hex conversion chart.

SECTION 4-3: LOGIC FUNCTIONS FOR BINARY NUMBERS

The Unidex 21 provides specific Logic functions for use with Binary format (Hexadecimal and Character) constants.

These functions are followed by a number (n). This number indicates the number of bytes (each byte contains 8 bits) that are effected by the function. (The Unidex 21 assigns four bytes of memory for each variable.) If no number follows a function, the default is 1, only Byte 1 will be effected by that function.

Bytes are arranged as follows:



SECTION 4-4: FUNCTIONS AND CONDITIONS

The following pages contain operations and conditions that may be utilized within the Unidex 21 Math package. All of these operations and conditions may also utilize User variables in addition to direct numeric entry.

+

NAME:

+ - Addition of Floating Point Numbers

FUNCTION:

The + function is used for addition of two floating point numbers.

FORMAT:

X = n+p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=20.2+40          ; Value of variable VAR1 will be 60.2.
VAR1=VAR2+VAR2        ; Value of variable VAR1 will be the sum of VAR2 and VAR3 or
                        60.2.

VAR2=H,28             ; Value of variable VAR2 will equal 40.

VAR1= BTF(VAR2)-20.2  ; Value of variable VAR1 will equal 60.2.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

NAME:

- - Subtraction of Floating Point Numbers

FUNCTION:

The - function is used to do subtraction of two floating point numbers.

FORMAT:

X=n-p

RETURNS:

The result is a floating point number.

EXAMPLE:

VAR1=40-20.2	; Value of variable VAR1 will be 19.8.
VAR1=VAR2-VAR3	; Value of variable VAR1will be the difference of VAR2 and VAR3 or 19.8.
VAR2=H,28	; Value of variable VAR2 will equal 40.
VAR1=BTF(VAR2)-20.2	; Value of variable VAR1 will equal 19.8.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

*

NAME:

* - Multiplication of Floating Point Numbers

FUNCTION:

The * function is used to do multiplication of two floating point numbers.

FORMAT:

X=n*p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2.2*4                ; Value of variable VAR1 will be 8.8.
VAR1=VAR2*VAR3            ; Value of variable VAR1 will be the product of VAR2 and VAR3
                            or 8.8.

VAR2=H,4                  ; Value of variable VAR2 will be 4.

VAR1=BTF(VAR2)*2.2        ; Value of variable VAR1 will be 8.8.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

/

NAME:

/ - Division of Floating Point Numbers

FUNCTION:

The / function is used for the division of two floating point numbers.

FORMAT:

X=n/p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2.0/4              ; Value of variable VAR1 will be 0.5.
VAR1=VAR2/VAR3          ; Value of variable VAR1 will be the quotient of VAR2 (dividend)
                          divided by VAR3 (divisor) or 0.5.

VAR2=H,4                ; Value of variable VAR2 will be 4.


VAR1=2.0/BTF(VAR2)      ; Value of variable VAR1 will be 0.5.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

( )

NAME:

( ) - Mathematical Phrases of Floating Point Numbers

FUNCTION:

The ( ) function is used to establish mathematical phrases of floating point numbers that are to be treated as a single term.

FORMAT:

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2*(2+4)              ; Value of variable VAR1 will be 12.
VAR1=VAR2*(VAR2+VAR3)     ; Value of variable VAR1 will the product of the quantity VAR2
                            plus VAR3 multiplied by VAR2 or 12.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

NAME:

! - Exponents of Floating Point Numbers

FUNCTION:

The ! function is used to express exponents of floating point numbers.

FORMAT:

X = n!p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2!4                ; Value of variable VAR1 will be 2x2x2x2 or 16.
VAR1=VAR2!VAR3          ; Value of variable VAR1, in this case, will be evaluated by
                          VAR2xVAR2xVAR2xVAR2 or 16.
```

Exponents can be some fixed number as well as the value of some other variable as shown in the example above.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

SIN

NAME:

SIN - Sine Value of a Floating Point Angle

FUNCTION:

The **SIN** function is used to derive the Sine value of a floating point angle.

FORMAT:

SIN(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=SIN(30.0)      ; Value of variable VAR1 will be the Sine of 30° or 0.5.
VAR1=SIN(VAR2)      ; Value of variable VAR1 will be the Sine of VAR2 or 0.5.
```

NOTES:

1) Variables may also be utilized.

2) All angles are expressed in decimal degrees.

RELATED COMMANDS:

DVAR

COS

NAME:

COS - Cosine Value of a Floating Point Angle

FUNCTION:

The **COS** function is used to derive the Cosine value of a floating point angle.

FORMAT:

COS(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=COS(60.0)      ; Value of variable VAR1 will be the Cosine of 60° or 0.5.
VAR1=COS(VAR2)      ; Value of variable VAR1 will be the Cosine of VAR2 or 0.5.
```

NOTES:

1) Variables may also be utilized.

2) All angles are expressed in decimal degrees.

RELATED COMMANDS:

DVAR

TAN

NAME:

TAN - Tangent Value of a Floating Point Angle

FUNCTION:

The **TAN** function is used to derive the Tangent value of a floating point angle.

FORMAT:

TAN(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=TAN(30.0)      ; Value of variable VAR1 will be the Tangent of 30° or 0.5773.
VAR1=TAN(VAR2)      ; Value of variable VAR1 will be the Tangent of VAR2 or 0.5773.
```

NOTES:

1) All angles are expressed in decimal degrees.

2) Variables may also be utilized.

RELATED COMMANDS:

DVAR

ATN

NAME:

ATN - Arctangent value of a Floating Point Number

FUNCTION:

The **ATN** function is used to derive the Arctangent (Inverse function of the Tangent) value of a floating point number.

FORMAT:

ATN(X)

RETURNS:

The result is a floating point number in decimal degrees.

EXAMPLE:

```
VAR1=ATN(1.732)      ; Value of variable VAR1 will be the Arctangent of 1.732 or 60°.
VAR1=ATN(VAR2)       ; Value of variable VAR1 will be the Arctangent of VAR2 or 60°.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

DEG

NAME:

DEG - Radian to Decimal Degree Conversion of Floating Point Numbers

FUNCTION:

The **DEG** function is used to convert radians to decimal degrees (both are floating point numbers).

FORMAT:

DEG(X)

RETURNS:

The result is a floating point number in decimal degrees.

EXAMPLE:

```
VAR1=DEG(0.5236 )      ; Value of variable VAR1 will be the decimal degree equivalent
                         of 0.5236 radians or 30°.
VAR1=DEG(VAR2)         ; Value of variable VAR1 will be the decimal degree equivalent
                         of VAR2 or 30°.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

RAD

NAME:

RAD - Decimal Degree to Radian Conversion of Floating Point Numbers

FUNCTION:

The **RAD** function is used to convert decimal degrees to radians (both are floating point numbers).

FORMAT:

RAD(X)

RETURNS:

The result is a floating point number in radians.

EXAMPLE:

```
VAR1=RAD(45.0)      ; Value of variable VAR1 will be the radian equivalent of 45° or
                      0.7854 radians.
VAR1=RAD(VAR2)      ; Value of variable VAR1 will be the radian equivalent of VAR2
                      or 0.7854 radians.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

ABS

NAME:

ABS - Absolute Value of a Floating Point Number

FUNCTION:

The **ABS** function is used to express the Absolute value of a floating point number.

FORMAT:

ABS(X)

RETURNS:

The result is a positive floating point number.

EXAMPLE:

```
VAR1=ABS(-30)          ; Value of variable VAR1 will be 30.0.
VAR1=ABS(VAR2)         ; Value of variable VAR1 will be the Absolute value of VAR2 or
                         30.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

SQR

NAME:

SQR - Square Root of a Floating Point Number

FUNCTION:

The **SQR** function is used to derive the Square Root value of a floating point number.

FORMAT:

SQR(X)

RETURNS:

The result is a positive floating point number.

EXAMPLE:

```
VAR1=SQR(25)        ; Value of variable VAR1 will be the Square Root of 25 or 5.0.
VAR1=SQR(VAR2)      ; Value of variable VAR1 will be the Square Root of VAR2 or
                      5.0.
```

NOTES:

1) Variables may also be utilized.

2) The Unidex 21 recognizes only positive floating point numbers.

RELATED COMMANDS:

DVAR

INT

NAME:

INT - Rounding of Floating Point Numbers

FUNCTION:

The **INT** function is used to round-off the fractional part of any floating point number.

FORMAT:

INT(X)

RETURNS:

The result is a floating point integer.

EXAMPLE:

```
VAR1=INT(123.05)      ; Value of variable VAR1 will be 123.0.
VAR1=INT(VAR2)        ; Value of variable VAR1 will round-off the fractional part of
                        VAR2 resulting in a value of VAR1 equals 123.0.

VAR3=INT(123.5)       ; Value of variable VAR1 will be 124.0.
VAR3=INT(VAR4)        ; Value of variable VAR3 will round-off the fractional part of
                        VAR4 resulting in a value of VAR4 equals 124.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

BTF

NAME:

BTF - Convert a Binary Number to a Floating Point Number

FUNCTION:

The **BTF** function is used to convert a binary number to a floating point number. Floating point numbers and binary numbers cannot be used within the same equation, unless one or the other is converted.

FORMAT:

BTF(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
(DVAR,VAR1,VAR2)          ; Define variables VAR1 and VAR2.

VAR1=H,1E                 ; Value of variable VAR1 will be 1E Hex (30.0).
VAR2=BTF(VAR1)            ; Value of variable VAR2 will be 30.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

FTB

NAME:

FTB - Convert a Floating Point Number to a Binary Number

FUNCTION:

The **FTB** function is used to convert a floating point number to a binary number. Floating point numbers and binary numbers cannot be used within the same equation, unless one or the other is converted.

FORMAT:

FTB(X)

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=40.0                ; Value of variable VAR1 will be H,28.
VAR2=FTB(VAR1)           ; Value of variable VAR2 will be converted to H,28.
```

NOTES:

1) Variables may also be utilized.

2) The fractional portion of the floating point number is rounded to the nearest integer.

RELATED COMMANDS:

DVAR

.EQ.

NAME:

.EQ. - Condition compares two Floating Point Numbers

FUNCTION:

The **.EQ.** function is used to compare two floating point numbers. If the numbers **are equal,** the result is "True" (1). If the numbers **are not equal,** the result is "False" (0).

FORMAT:

floating point number1**.EQ.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.EQ.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is equal to
                                SIN <30>.

VAR2=VAR1.EQ.SIN<30>          ; VAR2 equals H,1 if VAR1 is equal to SIN<30>, otherwise
                                VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.NE.

NAME:

.NE. - Condition compares two Floating Point Numbers

FUNCTION:

The **.NE.** function compares two floating point numbers. If the numbers **are not equal** the result is "True" (1). If the numbers **are equal**, the result is "False" (0).

FORMAT:

floating point number1**.NE.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.NE.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is not equal
                                to SIN <30>.

VAR2=VAR1.NE.SIN<30>          ; VAR2 equals H,1 if VAR1 is not equal to SIN<30>, otherwise
                                VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.GT.

NAME:

.GT. - Condition compares two Floating Point Numbers

FUNCTION:

The **.GT.** function compares two floating point numbers. If the value of the first number **is greater** than the value of the second number, the result is "True" (1). If the value of the first number **is not greater** than the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.GT.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

(JUMP,ENT1,VAR1**.GT.**SIN<30>)	; Program flow will go to Entry Point ENT1 if VAR1 **is greater** than SIN <30>.
VAR2=VAR1**.GT.**SIN<30>	; VAR2 equals H,1 if VAR1 **is greater** than SIN<30>, otherwise VAR2 equals H,0.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.GE.

NAME:

.GE. - Condition compares two Floating Point Numbers

FUNCTION:

The **.GT.** function compares two floating point numbers. If the value of the first number **is greater than or equal to** the value of the second number, the result is "True" (1). If the value of the first number **is not greater than or equal to** the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.GE.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.GE.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is greater
                                than or equal to SIN <30>.

VAR2=VAR1.GE.SIN<30>          ; VAR2 equals H,1 if VAR1 is greater than or equal to SIN
                                <30>, otherwise VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.LT.

NAME:

.LT. - Condition compares two Floating Point Numbers

FUNCTION:

The **.LT.** function compares two floating point numbers. If the value of the first number **is less than** the value of the second number, the result is "True" (1). If the value of the first number **is not less than** the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.LT.**floating point number2

RETURNS:

The result is a floating point number "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.LT.SIN<30>)     ; Program flow will go to Entry Point ENT1 if VAR1 is less than
                                  SIN <30>.

VAR2=VAR1.LT.SIN<30>            ; VAR2 equals H,1 if VAR1 is less than SIN<30>, otherwise
                                  VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.LE.

NAME:

.LE. - Condition compares two Floating Point Numbers

FUNCTION:

The **.LE.** function compares two floating point numbers. If the value of the first number **is less than or equal to** the value of the second number, the result is "True" (1). If the value of the first number **is not less than or equal to** the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.LE.** floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.LE.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is less than
                                or equal to SIN <30>.

VAR2=VAR1.LE.SIN<30>          ; VAR2 equals H,1 if VAR1 is less than or equal to SIN<30>,
                                otherwise VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.NOTn.

NAME:

.NOTn. - Converts Binary Numbers

FUNCTION:

The **.NOTn.** function converts each bit from one to zero or zero to one.

FORMAT:

.NOTn. Binary number ; Where n is the number of bytes to invert, starting with the least significant byte.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=.NOT2.H,F32        ; Bytes 1 and 2 are "inverted".
                        ; Hex F32 converted to Binary  is:
                        ;      B2              B1
                        ;  00001111          00110010
```

once complemented it becomes Hex F0CD which is:

```
                        ;      B2              B1
                        ;  11110000          11001101

VAR1=.NOT1.H,F32        ; Byte 1 is "inverted".
                        ; Hex F32 converted to Binary is:
                        ;      B2              B1
                        ;  00001111          00110010
```

once byte 1 is complemented it becomes Hex FCD which is:

```
                        ;      B2              B1
                        ;  00001111          11001101
```

Note that byte 2 was not converted in this case.

.NOTn.

NOTES:

1) Variables may also be utilized.

2) This function requires only one operand.

RELATED COMMANDS:

DVAR

.ANDn.

NAME:

.ANDn. - AND's two Binary Numbers

FUNCTION:

The **.ANDn.** function is used to "AND" two binary numbers. The table below illustrates the **.ANDn.** function:

1 **.AND.** 1 = 1
1 **.AND.** 0 = 0
0 **.AND.** 1 = 0
0 **.AND.** 0 = 0

The "AND" operation is performed on a bit-by-bit basis, as illustrated above.

FORMAT:

Binary number.**ANDn.**Binary number ; Where "n" is the number of bytes to "AND", starting with the least significant byte.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.AND2.VAR2     ; Value of variable VAR1 is equal to the value of VAR1 .AND.
                          VAR2.

                        ; If   VAR1 is H,8A or 10001010
                               VAR2 is H,9C or 10011100
                          then VAR1 equals H,88 or 10001000.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.ORn.

NAME:

.ORn. - OR's two Binary Numbers

FUNCTION:

The **.ORn.** function is used to "OR" two binary numbers. The table below illustrates the **.ORn.** function:

1 **.ORn.** 1 = 1
1 **.ORn.** 0 = 1
0 **.ORn.** 1 = 1
0 **.ORn.** 0 = 0

The "OR" operation is performed on a bit-by-bit basis, as illustrated above.

FORMAT:

Binary number**.ORn.**Binary number ; Where "n" is the number of bytes to "OR", starting with the least significant byte.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.OR.VAR2      ; Value of variable VAR1 is equal to the value of VAR1 .OR.
                         VAR2.

                       ; If   VAR1 is H,8A or 10001010
                              VAR2 is H,9C or 10011100
                        then VAR1 equals H,9E or 10011110.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.XORn.

NAME:

.XORn. - EXCLUSIVE OR'S two Binary Numbers

FUNCTION:

The **.XORn.** function is used to "EXCLUSIVE OR" two binary numbers. The table below illustrates the **.XORn.** function:

1 **.XORn.** 1 = 0
1 **.XORn.** 0 = 1
0 **.XORn.** 1 = 1
0 **.XORn.** 0 = 0

The "XOR" operation is performed on a bit-by-bit basis, as illustrated above.

FORMAT:

Binary number1**.XORn.**Binary number2 ; Where "n" is the number of bytes to "EXCLUSIVE OR", starting with the least significant byte.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.XOR.VAR2      ; Value of variable VAR1 is equal to the value of VAR1 .XOR.
                          VAR2.

                        ; If   VAR1 is H,8A or 10001010
                               VAR2 is H,9C or 10011100
                          then VAR1 equals H,9G or 00010110.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.LSLn.

NAME:

.LSLn. - Logical Shift Left of Binary Numbers

FUNCTION:

The**.LSLn.** (Logical Shift Left) shifts a specified number of bits left. It requires two operands in a specified order.

FORMAT:

Binary number**.LSLn.**Binary number

The first operand provides the number being shifted. The second operand specifies the number of bits of the shift, and "n" indicates the number of bytes to shift across.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.LSL.H,3      ; The Bits in Byte 1 are shifted to the left 3 spaces.

                       ; If Byte 1 is 10101010 the first 3 bits on the left (101) will be
                         replaced by the next 3 (010). The empty spaces on the right
                         result will be replaced with zeros. The result will be
                         01010000.

VAR1=VAR1.LSL2.H,3     ; The 3 bits to be shifted left from Byte 1 will shift over to Byte
                         2. The 3 bits shifted out of Byte 2 will "fall off". The 3 empty
                         spaces on the far right of Byte 1 will be replaced with zeros.
```

B4	B3	B2	B1
00000110	00100011	10010111	01101111

Following VAR1=VAR1.**LSL**2.H,3 VAR1 will be:

B4	B3	B2	B1
00000110	00100011	10111011	01111000

.LSLn.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.LSRn.

NAME:

.LSRn. - Logical Shift Right of Binary Numbers

FUNCTION:

The**.LSRn.** (Logical Shift Right) function shifts a specified number of bits right. In all other respects it is the same as the **.LSLn.** function.

FORMAT:

RETURNS:

EXAMPLE:

NOTES:

RELATED COMMANDS:

.ROLn.

NAME:

.ROLn. - Rotate Binary Numbers Left

FUNCTION:

The**.ROLn.** (Rotate Left) function shifts a specified number of bits left and moves the displaced bits to the far right. It requires two operands in a specified order.

FORMAT:

Binary number**.ROLn.**Binary number

The first operand provides the number being rotated. The second operand specifies the number of bits for the rotation, and "n" indicates the number of bytes to rotate across.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.ROL.H,3      ; The Bits in Byte 1 are shifted to the left 3 spaces and the
                         displaced Bits are "wrapped around" to the right. If Byte 1 is
                         10101010 the first 3 Bits (101) will be replaced by the next
                         three (010). The empty spaces on the right are filled with the
                         displaced Bits resulting in 01010101.

VAR1=VAR1.ROL2.H,3     ; The 3 Bits to be shifted left from Byte 1 will shift over to Byte
                         2. The displaced Bits from Byte 2 will "wrap around" to the
                         right, replacing the empty spaces in Byte 1.
```

B4	B3	B2	B1
00000110	00110111	01110010	11001010

Following VAR1=VAR1.**ROL2**.H,3 VAR1 will be:

B4	B3	B2	B1
00000110	00110111	10010110	01010011

.ROLn.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.RORn.

NAME:

.RORn. - Rotate Binary Numbers Right

FUNCTION:

The**.RORn.** (Rotate Right) function shifts a specified number of bits right. The displaced bits replace the empty spaces on the far left. In all other respects it is the same as the **.ROLn.** function.

FORMAT:

RETURNS:

EXAMPLE:

NOTES:

RELATED COMMANDS:

.ADDn.

NAME:

.ADDn. - Addition of Binary Numbers

FUNCTION:

The **.ADDn.** function is used to do addition of two binary numbers.

FORMAT:

Binary number1**.ADDn.**Binary number2 ; Where "n" indicates the number of bytes to add.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.ADD.VAR2      ; Variable VAR1 is added to variable VAR2.


 10111011               ; Binary numbers are added in the conventional way
+01001011                 with the "carrying" of numbers. When ADDing binary
 1|00000110               numbers in an 8 Bit register, the last carried number
                          is dropped.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.SUBn.

NAME:

.SUBn. - Subtraction of Binary Numbers

FUNCTION:

The **.SUBn.** function is used to do subtraction of two binary numbers.

FORMAT:

Binary number1**.SUBn.**Binary number2 ; Where "n" indicates the number of bytes to subtract.

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=VAR1.SUB.VAR2          ; Variable VAR2 is subtracted from variable VAR1.

 10111011                   ; Binary numbers are subtracted in the conventional
-01001011                     way with the "borrowing" of numbers.
 01110000

VAR1=VAR2.SUB.VAR1          ; Value of variable VAR1 will be the difference between VAR2
                              and VAR1 or 10010000.

 01001011                   ;
-10111011
 10010000
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.TSTA.

NAME:

.TSTA. - Test And of Binary Numbers

FUNCTION:

The **.TSTA.** (Test And) function tests several bits and produces a "True " (1) result only if all bits are tested true. This function requires two operands in a specified sequence.

FORMAT:

Binary number1**.TSTA.**Binary number2

The first operand provides the number being tested. The second operand specifies the bits that are to be tested.

RETURNS:

The result is a binary number, "1" for True, "0" for False.

EXAMPLE:

```
VAR1=H,3F.TSTA.H,11          ; H,3F = 0011 1111
                             ; H,11 = 0001 0001
                             ; VAR1 equals H,1 - both Bits tested true (1)

(JUMP,ENT1,$INP.TSTA.H,12.EQ.H,1)   ; $INP = 0011 1010
                                    ; H,12 = 0001 0010
```

Both Bits tested true, therefore, the jump to ENT1 will be accomplished.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.TSTO.

NAME:

.TSTO. - Test Or of Binary Numbers

FUNCTION:

The **.TSTO.** (Test Or) function tests several bits and produces a "True" (1) result if any of the bits test true. This function requires two operands in a specified sequence.

FORMAT:

Binary number1**.TSTO.n.**Binary number2

The first operand provides the number being tested. The second operand specifies the number of bits to be tested.

RETURNS:

The result is a binary number, "1" for True, "0" for False.

EXAMPLE:

```
VAR1=H,35.TSTO.H,3              ; H,35 = 0011 0101
                                ; H,3 = 0000 0011
                                ; VAR1 equals H1, one Bit tested true (1).

(JUMP,ENT1,$001.TSTO.H,6.EQ.H,1)  ; $001 = 0010 0101
                                ; H,6 = 0000 0110
```

One Bit tested true, therefore, the jump to ENT1 will be accomplished.

```
$023=H,3F.TSTO.H,C0             ; H,3F = 0011 1111
                                ; H,C0 = 1100 0000
```

Data on I/O port $023 will be set to zero, the result is false and no Bits tested are true.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.HI.

NAME:

.HI. - Compare unsigned Binary Numbers

FUNCTION:

The **.HI.** function compares two unsigned binary numbers. The "Most Significant Bit" (MSB) is compared first. If both numbers have the same value (either a 0 or a 1) the next bit of both the numbers is compared. If the first number has a bit value of 1 and the second number has a bit value of 0, the result would be "True" (1), otherwise, the result is "False" (0).

FORMAT:

Binary number1**.HI.**Binary number2

RETURNS:

The result is a binary number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.HI.H,30)      ; Program flow will go to Entry Point ENT1 if variable VAR1 is
                                greater than H,30.

VAR2=VAR1.HI.H,30             ; VAR2 equals H,1 if VAR1 is greater than H,30 otherwise,
                                VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.LS.

NAME:

.LS. - Compares unsigned Binary Numbers

FUNCTION:

The **.LS.** function compares two unsigned binary numbers. The "Most Significant Bit" (MSB) is compared first. If both numbers are of equal value (either a 0 or a 1) or the first number has a value of 0 and the second number has a value of 1 the result is "True" (1), otherwise, the result is "False" (0).

FORMAT:

Binary number1**.LS.**Binary number2

RETURNS:

The result is a binary number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.LS.H,30)      ; Program flow will go to Entry Point ENT1 if the Bits of VAR1
                                are equal to or less than H,30.

VAR2=VAR1.LS.H,30             ; VAR2 equals H,1 if the Bits of VAR1 are equal to or less than
                                H,30, otherwise VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

CHAPTER 5: SUMMARY

COMMAND SUMMARY:

The following list of commands may be used as a quick reference. Detailed information for each of the commands is available in Chapter 3 of this manual.

%	Program Title
/	Block Delete Operator
;	Comment Operator
$HSI	High Speed Interrupt Buffer
$INn	Input System Variable
$MFO	Manual Feed Override System Variable
$nAP	Absolute Position Register Variables
$nRP	Relative Position Register Variables
$OTn	Address Logic Output System Variable
$POT	RS-232 Port Receive Buffer
$R	Read command
$RTP	Real Time Position Buffer
$TOD	Time of Day System Variable
$000-$FAF	Extended Input/Output Capabilities
ABTS	Abort Subroutine
ACDE	Maximum Acceleration/Deceleration
AFCO	Auto-Focus
CCP	Cutter Compensation
CLS	Call Subroutine
COEF	Coefficient for Parabolic Trajectory Ramping
COMM	Communication through RS-232 Port A/B
CPAG	Customer Display Page
DARY	Define Array
DENT	Define Entry Point
DFLS	Define Library Subroutine
DFS	Define Subroutine
DRUN	Dry Run
DVAR	Define Variable

COMMAND SUMMARY (CON'T):

DZON	Define Safe Zone
ELPS	Ellipse Look-Up Table
F	Define Axis Feedrate
FILT	Digital Filter
FREE	Axis Free Run (enabled or disabled)
FXOF	Fixture Offset
G0	Point-to-Point Positioning at rapid traverse rate
G1/G11/H1/H11	Linear Contouring
G2/G12/H2/H12	CW Circular Contouring
G3/G13/H3/H13	CCW Circular Contouring
G4	Dwell
G5/G15/H5/H15	Three Point Interpolation of two dimensional arc for contouring
G8	Velocity Profiling (acceleration)
G9	Velocity Profiling (deceleration)
G17/H17	Axis Plane Designation for execution of Circular Interpolation
G18/H18	Axis Plane Designation for execution of Circular Interpolation
G19/H19	Axis Plane Designation for execution of Circular Interpolation
G23	Corner Rounding
G24	Non-Corner Rounding
G40	Deactivates Cutter Radius Compensation
G41	Activates Cutter Radius Compensation-Left
G42	Activates Cutter Radius Compensation-Right
G70	English Programming
G71	Metric Programming
G90	Absolute Position Programming
G91	Relative Position Programming
G92	Software Home
HOME	Hardware Home
HSIE	High Speed Interrupt Control

COMMAND SUMMARY (CON'T):

HWEL	Handwheel Control
I,J,K,P and i,j,k,p	Circular Interpolation parameters
INT1/INT2	User Interrupt Control
JOIN	Join Subprogram to Main Program
JUMP	Jump to User Defined Entry Block
LCD, OAB and lcd, oab	Polar coordinates for Circular Interpolation
LIMT	Set Software Limit
LINK	Link Multiple Axes to a Group
MALC	User's Memory Allocate
MIR	Mirror Image
MORG	Machine Origin
MSG	Display Screen Messages
M, S, T	Output Capability (16 bit outputs)
MSTD	MST Strobe/Ack Delay, also Output during INT1/INT2, Option 3
MTOR	Motor Current Command Control
OPEN, MEND, END, WRIT	Reads/Writes data to memory
PID	Sets Kp, Ki, Kd, Kf1, and Kf2 Values
PLAY	Play Back the Recorded Axes Motion
PLC	Programmable Logic Control Interface
PLNE	Define Circular Contour Plane
POP	Pull Data Out of User's Stack
PORT	RS-232 Port Background Data Collection
PUSH	Push Data Into User's Stack
RAMP	Define Axis Ramping time
RECO	Record Axis Motion
REF	Send Axis to Hardware Home
RETP	Real Time Position Fetch
ROTA	Parts Rotation
RPT	Repeat Loop
RTRS	Retrace
SCF	Scaling Factor
SCO	Scaling ON/OFF Control

COMMAND SUMMARY (CON'T):

SCRB	Character Scribing
SIOC	System I/O Control
SKEY	Define Custom Softkey
SKLV	Change Custom Softkey level
SKYD	Define Softkey table
SLEW	Joystick/Teach Pendant/Trackball/Mouse Slew
SYNC	Synchronize Two Unidex 21's
STKP	User's Stack Pointer Adjust
TERM	Terminal Display for MSG function
TRAJ	Axis Acceleration/Deceleration Trajectory, Type Selection
TRAK	Position Tracking Display Control
UMFO	User Defined MFO Setting
ZONE	Define Safe Zone

COMPARISON OPERATORS SUMMARY:

.EQ.	Equal
.NE.	Not Equal
.GT.	Greater Than
.GE.	Greater Than or Equal To
.LT.	Less Than
.LE.	Less Than or Equal To

FLOATING POINT OPERATORS SUMMARY:

+	Addition Operator
-	Subtraction Operator
*	Multiplication Operator
/	Division Operator
()	Operation Precedence
!	Exponentials
SIN	Sine Value
COS	Cosine Value

FLOATING POINT OPERATORS SUMMARY (CON'T):

TAN	Tangent
ATN	Arctangent
DEG	Radian to Degree Conversion
RAD	Degree to Radian Conversion
ABS	Absolute Value
SQR	Square Root
INT	Float to Integer Conversion (Rounding)
FTB	Float to Binary

BINARY OPERATORS SUMMARY:

.NOTn.	1's Complement
.ANDn.	And
.ORn.	Or
.XOR.	Excluive or
.LSLn.	Logical Shift Left
.LSRn.	Logical shift Right
.ROLn.	Rotate Left
.RORn.	Rotate Right
.ADDn.	Addition
.SUBn.	Subtraction
.TSTA.nnnn.	Test And
.TSTO.nnnn.	Test or
.HI.	Unsigned Comparison Higher
.LO.	Unsigned Comparison Lower
BTF	Binary to Float Conversion

SYSTEM VARIABLES SUMMARY:

$URP, $XRP, $YRP, $ZRP	Relative Position Variables
$uRP, $xRP, $yRP, $zRP,	
$UAP, $XAP, $YAP, $ZAP	Absolute Position Variables
$uAP, $xAP, $yAP, $zRP	

SYSTEM VARIABLES SUMMARY (CON'T):

$TOD	Time Of Day
$INn	Current Input(s)
$OTn	Output(s) (Write only)
$MFO	Manual Feed Override
$R	Read Command
$HSI	High Speed Interrupt Buffer
$RTP	Real Time Position Buffer
$POT	RS-232 Port Receive Buffer
$000-$7FF	Motorola I/O Channels
$800-$F6F	PLC Dual Port Memory Locations
$F70-$FAF	PAMUX I/O Channels

APPENDIX 1: CUTTER COMPENSATION (ICRC)

ICRC stands for Intersectional Cutter Radius Compensation. In cutting a workpiece, sometimes the radius of the cutter must be taken into consideration. For example, when an endmill is used to cut the sides of a workpiece, the center of the endmill follows the programmed path. The outside edge of the endmill cuts around the actual workpiece offset from the programmed path by the radius of the tool.

Cutter radius compensation is an option which allows the operator to program the center of the cutter in such applications, so that the outside edge of the endmill cuts along the programmed path. Without this option the operator would have to offset the actual piece dimensions with the radius of the tool. When it becomes necessary to program angles other than 90°, it is no longer just a radius offset. This option dramatically decreases the programming effort by handling all the axis offsets. Also, the **ICRC** option allows the same program to be used with tools of different diameters just by changing the tool diameter information with the **CCP** command.

COMMANDS:

There are 4 commands to use:

G40	- Cutter compensation/Offset, cancel
G41	- Cutter compensation-Left, turn ON
G42	- Cutter compensation-Right, turn ON
(CCP,AX1,AX2,DIAM)	- Axis pair and tool diameter

The commands **G41** and **G42** are oriented in the direction of cutter motion. Example:

G41	Causes the cutter to make a path to the left of the nominal path by the amount of the radius determined from the tool diameter. Left is relative to the direction in which the cutter is moving.
G42	Causes the cutter to make a path to the right of the nominal path by the amount of the radius determined from the tool diameter. Right is relative to the direction in which the cutter is moving.

COMMANDS (CON'T):

The following diagram illustrates when to choose the **G41** or **G42** command.

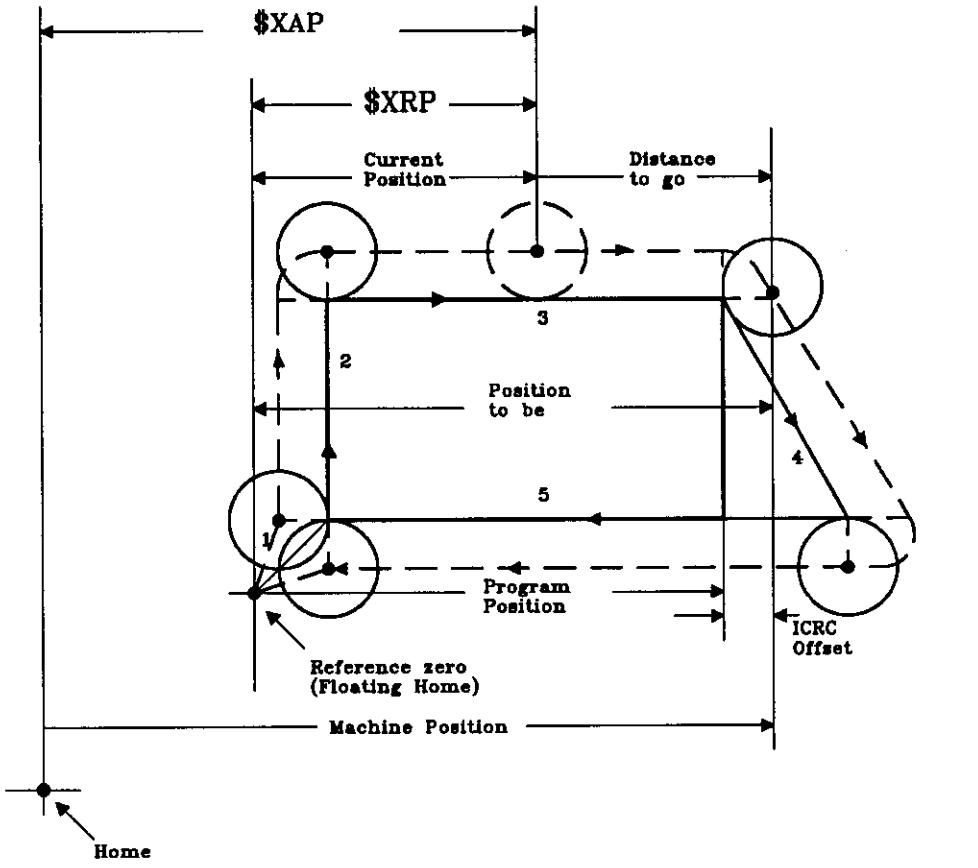
START-UP:

To start **ICRC**, first define the axis pair and tool diameter using the **CCP** command. Then enter a **G41** or **G42** followed by a start-up move before starting a cut. At this point, the operator can program the center of the tool to follow the workpiece contour.

As you can see, the solid line is the shape to be cut. The cutter center traces the dotted line, which is offset from the true shape of the workpiece by the amount of the radius.

The following example shows all X position registers, assuming the Unidex 21 is in the middle of move #3 at the dashed circle position. It also demonstrates the tool adjustments necessary to accommodate **ICRC**.

START-UP (CON'T):



When making a linear to linear move using **ICRC**, you may switch from **G41** to **G42** and vice versa with no problem.

For the sake of accuracy however, when making a linear to circular or circular to linear move and switching from **G41** to **G42**, a transitional move containing a **G40** (cancel **ICRC**) must be placed between the two.

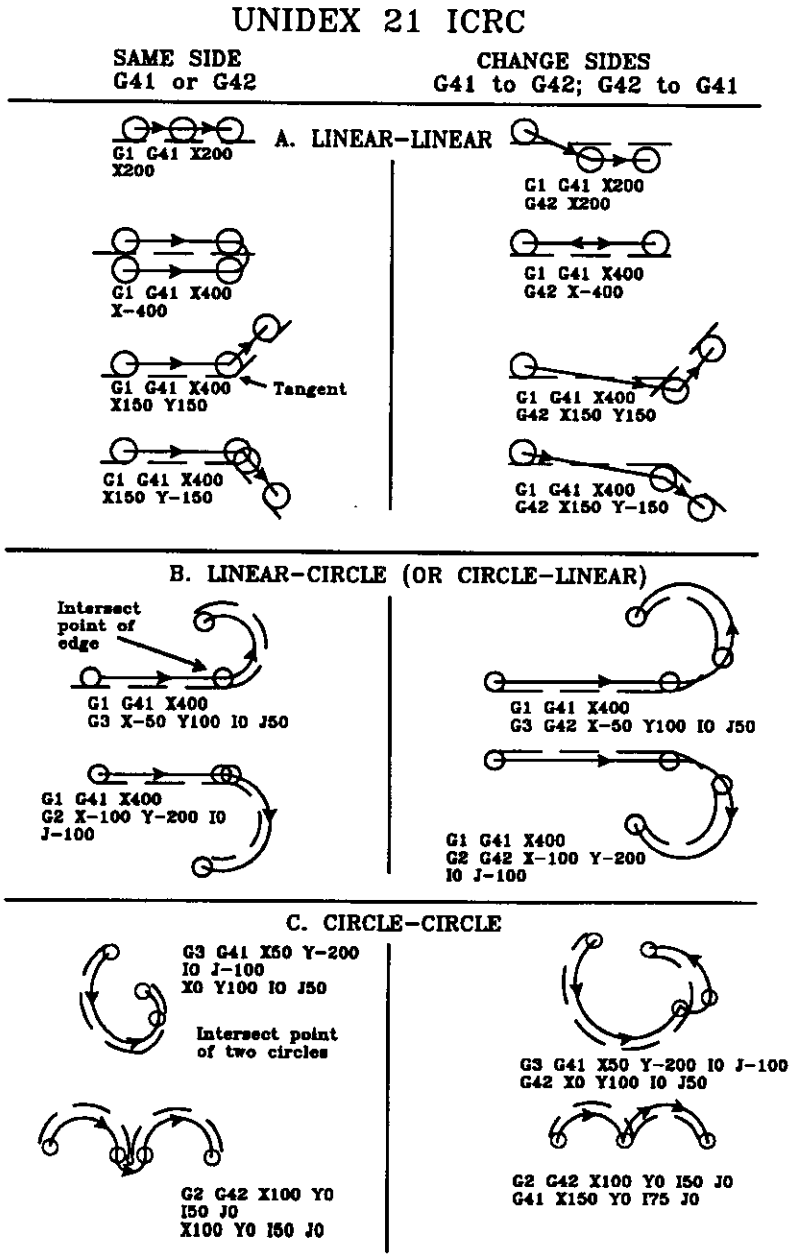
Another method of switching sides with **ICRC** is to break the transitional move (the one between linear and circular) into two moves. Execute one in **G41** and one in **G42**.

CANCELING ICRC:

To end **ICRC**, make an ending move to direct the cutter away from the workpiece after cutting, using the **G40** command to turn off the **ICRC**.

ICRC must be canceled before an **M2** command.

ICRC CONFIGURATIONS:



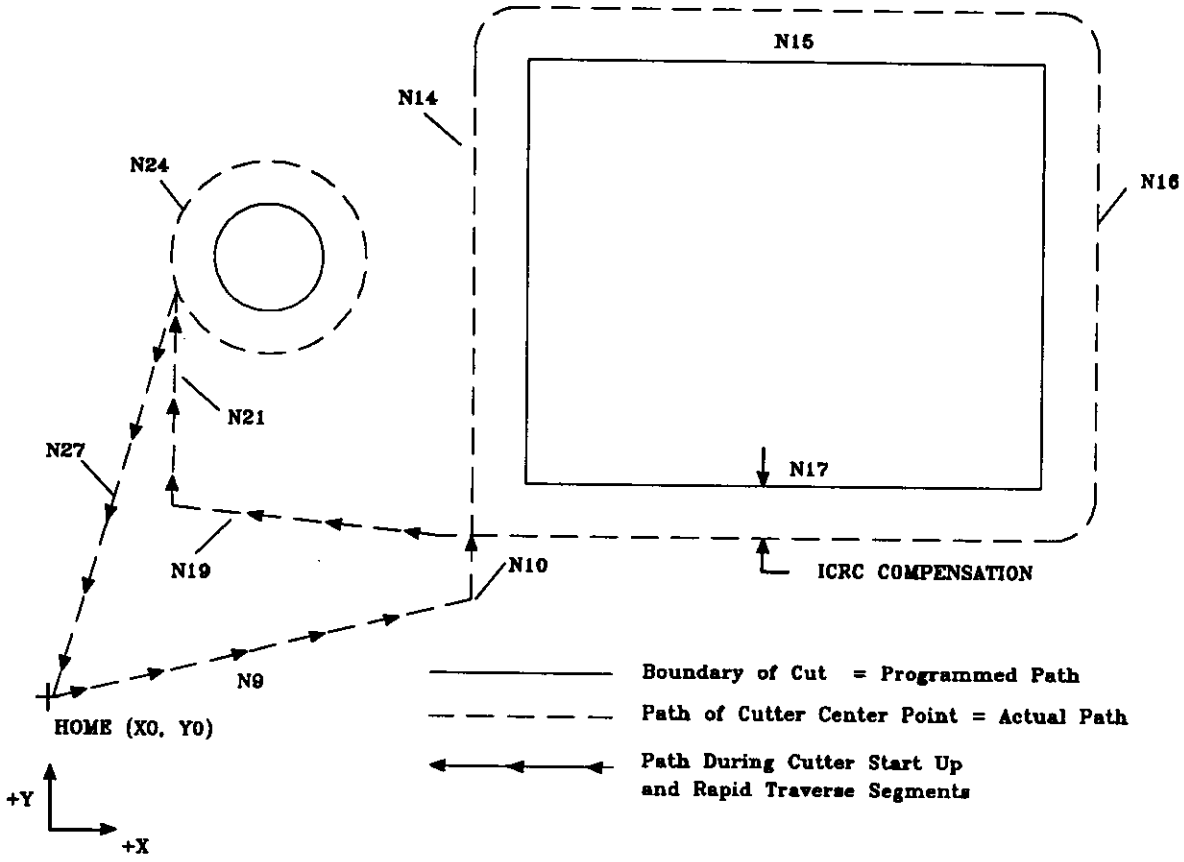
PROGRAMMING EXAMPLE:

The following Unidex 21 program cuts a rectangle and a circle from a material using a laser. Radius compensation **ICRC** is used to compensate for laser beam "kerf" (the width of the cut made by a tool such as a saw, torch, laser beam, or water jet). The **ICRC** compensation offsets are exaggerated for clarity.

An X-Y positioning stage combination is used to move the part under the cutting laser. Distances are in inches.

LINE	COMMAND	DESCRIPTION
N2	M80	Laser is OFF.
N3	(REF,X,Y)	Find Hardware Home for the X and Y axes.
N6	G91 G70	Initiate Incremental Positioning and English Programming (inches).
N7	(CCP, X, Y, 0.5)	Set ICRC to X/Y plane with a tool diameter of 0.5 inches.
N8	G41	Turn ON tool radius compensation (ICRC). Compensate to the left of the workpiece.
N9	G1 X4.25 Y0.6 F100.0	Linear Offset Move for the X and Y axes.
N10	Y0.9	Y offset to start of rectangle pattern.
N11	M90	Laser is ON.
N12	(MSG, Cutting Rectangle)	A message will be displayed.
N13	G4 F0.5	A Dwell of 0.5 seconds is established.
N14	Y3.1	Side of rectangle
N15	X3.4	Side of rectangle
N16	Y-3.1	Side of rectangle
N17	X-3.4	Side of rectangle
N18	M80	Laser is OFF.
N19	G40 X-2.75	Deactivate the Cutter Radius Compensation and perform the end move.
N20	G41	Activate the Cutter Radius Compensation left.
N21	X0.5 Y1.5	Offset to the starting point of the circle.
N22	M90	Laser is ON.
N23	(MSG,0, Cutting Circle)	A message will be displayed.
N24	G2 I0.6 J0	A 0.6 inch radius circle will be produced.
N25	M80	Laser is OFF.
N26	G4 F0.5	A Dwell of 0.5 seconds is established.
N27	G40 G90 G0 X0.0 Y0.0	Go home at a rapid traverse rate.
N29	M2	End of The Program

PROGRAMMING EXAMPLE (CON'T):



APPENDIX 2: CORNER ROUNDING/VELOCITY PROFILING

CORNER ROUNDING (G23) AND NON-CORNER ROUNDING (G24) MODES:

An insertion of the code **G23** within a motion command statement (e.g., G8 G1 G23 X100.0 Y200.0) or on a separate line, sets the Unidex 21 for the "corner rounding" mode. The definition of "corner rounding" as applied to the Unidex 21 is as follows:

- Set the trajectory generator of the Unidex 21 to execute all subsequent motion commands (either linear or circular commands, in a single or multi-axis configuration) at the commanded feedrate. Depending upon the setting of Main System Parameter #60 or 61, the Unidex 21 may either utilize the ramptime parameter setting (**RAMP**) or digital filter (**FILT**) for acceleration/deceleration.

- Disable the "motion complete flag" in the servo control loops of each of the enabled axes. This effectively tells the trajectory generator to "not wait" for the given axes to stabilize on the final commanded position of the just completed motion command before sending the next motion command(s) to the servo loops.

An insertion of the code **G24** instead of the code **G23** "inverts" the meaning of the two statements for the **G23** mode. In otherwords with **G24** inserted, the "motion complete flag" is enabled insuring that the commanded position of the currently executing motion command is completely executed by the servo loops before the next command(s) is sent down by the trajectory generator.

Also, it is possible to utilize both the axes **RAMP** or **FILT** commands (or their related parameter settings) in order to optimize the trajectory command execution and provide similar performance to "following error" based motion controllers. Please refer to related commands and parameters for further explanation.

Note that commands **G23** and **G24** are system modal. In other words, once the command **G23** is encountered either in the execution of a program or the system "MDI" (manual) mode it stays in effect until changed by **G24** or a system reset.

PROFILING (G8) AND NON-PROFILING (G9) MODES:

An insertion of the **G8** command enables the velocity profiling mode. Unlike the **G23/G24** commands which are modal, this command must be inserted within each motion command in order for it to be recognized for the given motion command (e.g., G8 G1 X100.0 Y100.0). The definition of "profiling" as applied to the Unidex 21 is as follows:

- Each motion command processed by the trajectory generator is setup to link the moves together such that constant vector velocity is maintained between commanded motion blocks or, if the Feedrate is changed, the system will accelerate/decelerate to the next velocity utilizing either **RAMP** time, **FILT** value, or the combination of both. Note that the two aforementioned commands have user program setting capability or Main System Parameter settings. Refer to related commands and parameters for further explanation.

- If the motion command containing this code is the first motion command within a string of other **G8** motion commands (e.g., if it is the first command after a previously executed **G9** command), the vector velocity of this command will begin at zero speed. Subsequent **G8** commands encountered after this first **G8** command will be executed beginning at the vector velocity reached by the previous **G8** command, if the Feedrate is not changed in this program block.

- If the **G24** mode is enabled, each new vector velocity specified by **G8** commands will be "ramped" to that specified velocity starting at the "attained" velocity of the previous **G8** command (the ramp time being specified by the **RAMP** command, e.g., (RAMP,100)). Note that for this particular mode of operation, it is important that the vector velocity "direction" be maintained between ending of one **G8** command to the beginning of the next **G8** or ending **G9** command in order for the desired position trajectory to be maintained. If vector velocity direction needs to be changed, it MUST be accomplished through the programming of an "arc" or circle segment. This is the only way to ensure tangency.

- If the **G23** mode is enabled and Main System Parameter #61 is set to NO, each new vector velocity specified by **G8** commands will be instantaneously executed at the specified velocity. In this mode, the **RAMP** command is ignored by the trajectory generator. This means that when it is desired to change vector velocity directions between **G8** programmed segments, the programmer need only specify new "linear" command segments (e.g., G8 G1 X100.0 Y100.0) in order to change direction. The acceleration/deceleration can only be provided by the digital filter setting (either via **FILT** command or Main System Parameter #60).

PROFILING (G8) AND NON-PROFILING (G9) MODES (CON'T):

An insertion of the **G9** command disables the velocity profiling mode and essentially puts the trajectory generator in the "point to point" position mode. The motion command executed under this mode is "guaranteed" to feed the servo control loop(s) the final or ending position of the specified command. Note that this command is "inferred" (e.g., considered the default command) in the **G8/G9** mode of operation. The motion command "G1 X100.0 Y100.0" is automatically recognized by the trajectory generator software as being the command "G9 G1 X100.0 Y100.0". The definition of "non-profiling" as applied to the Unidex 21 is as follows:

- Each **G9** motion command processed by the trajectory generator is set up to execute based on the priority of attaining the final commanded position. The velocity profile will be calculated in such a fashion as to insure that the position generated by the trajectory generator is guaranteed to be obtained at a final axis velocity of zero.

- A **G9** command NOT preceded by a **G8** command has a beginning and ending vector velocity of zero.

- A **G9** command preceded by a **G8** command has a beginning vector velocity equal to the ending velocity of the previous **G8**.

In the "following error" mode, an inherent "exponential" ramp function, is added to each of the servo axis, effectively "smoothing" out sudden changes in velocity commands. The effect is illustrated in Figure A-1.

PROFILING (G8) AND NON-PROFILING (G9) MODES (CON'T):

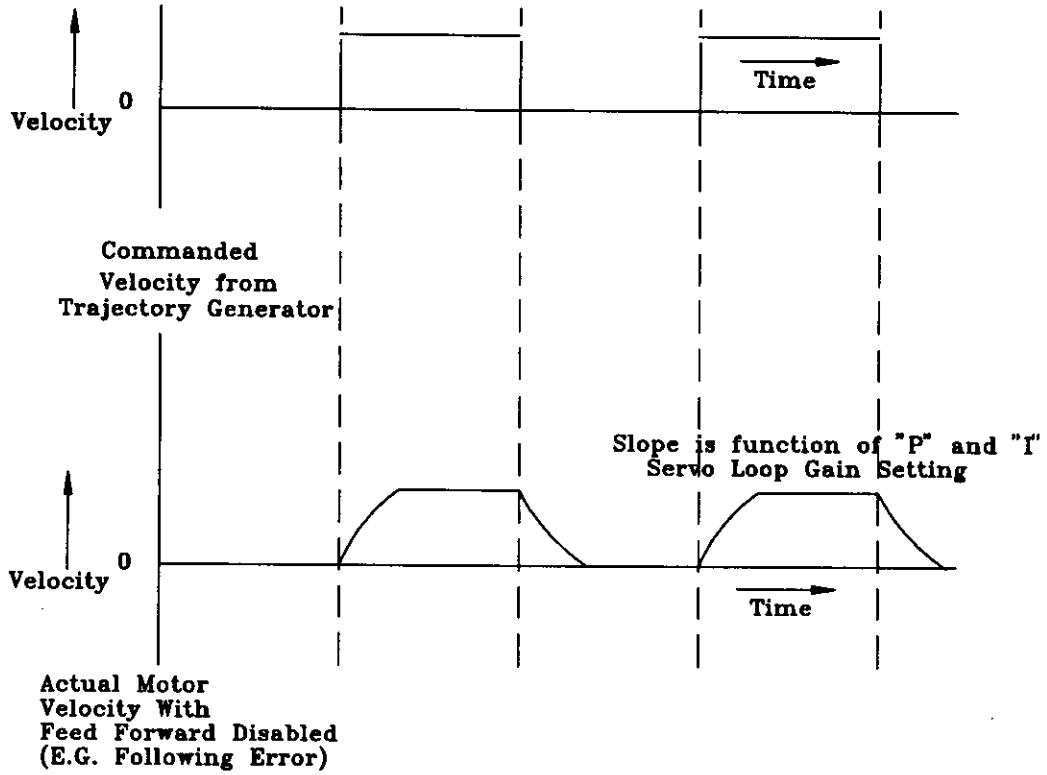


Figure A-1: Effects of "G23" Mode on Trajectory Generators with Following Error Enabled (KF1=0)

This effect may also be obtained using the **FILT** command or Main System Parameter #60. In this mode of operation, however, the exponential ramp is applied to the velocity command. Note that following error does not need to be enabled in this mode.

PROFILING (G8) AND NON-PROFILING (G9) MODES (CON'T):

Combining the **G8** function with **G23**, and enabling "following error" or using the **FILT** command for the servo loops allows the position profiles such as the one depicted in Figure A-2, to be executed.

Referring to Figure A-2, take note that the joining points of the **G8** command segments (which are strings of arc and linear commands) are not tangent to each other. When operating in the **G23** mode, segment tangency is not required. In addition, velocities for each segment can be changed at will (changing of velocities can not be made apparent in this diagram) without significantly changing the intended profile.

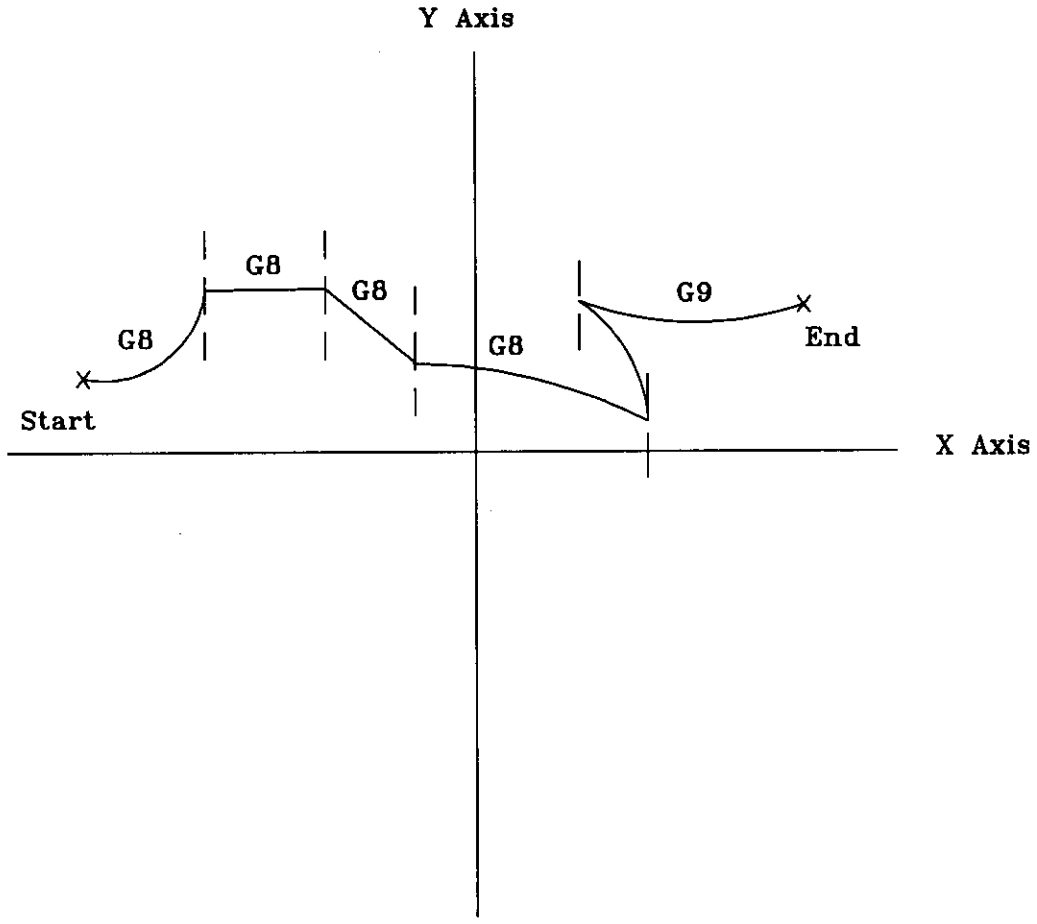


*Figure A-2: Typical Two Axis Position Profile Using **G8**, **G23** with Following Error or Filt Enabled*

PROFILING (G8) AND NON-PROFILING (G9) MODES (CON'T):

Combining the **G8** function with **G24** allows position profiles such as the one depicted in Figure A-3 to be executed. Note that for this particular mode of operation, allowing following error to be enabled or disabled produces negligible effects on the motion profile unless high vector velocities are commanded.

Referring to Figure A-3, notice that the joining points of each segment are tangent. Tangency between all **G8** and **G9** segments is a requirement for **G24** operation. As in the case of the **G23** operation described above, segment velocities can be changed at will. In addition, ramptime can be changed from segment to segment.

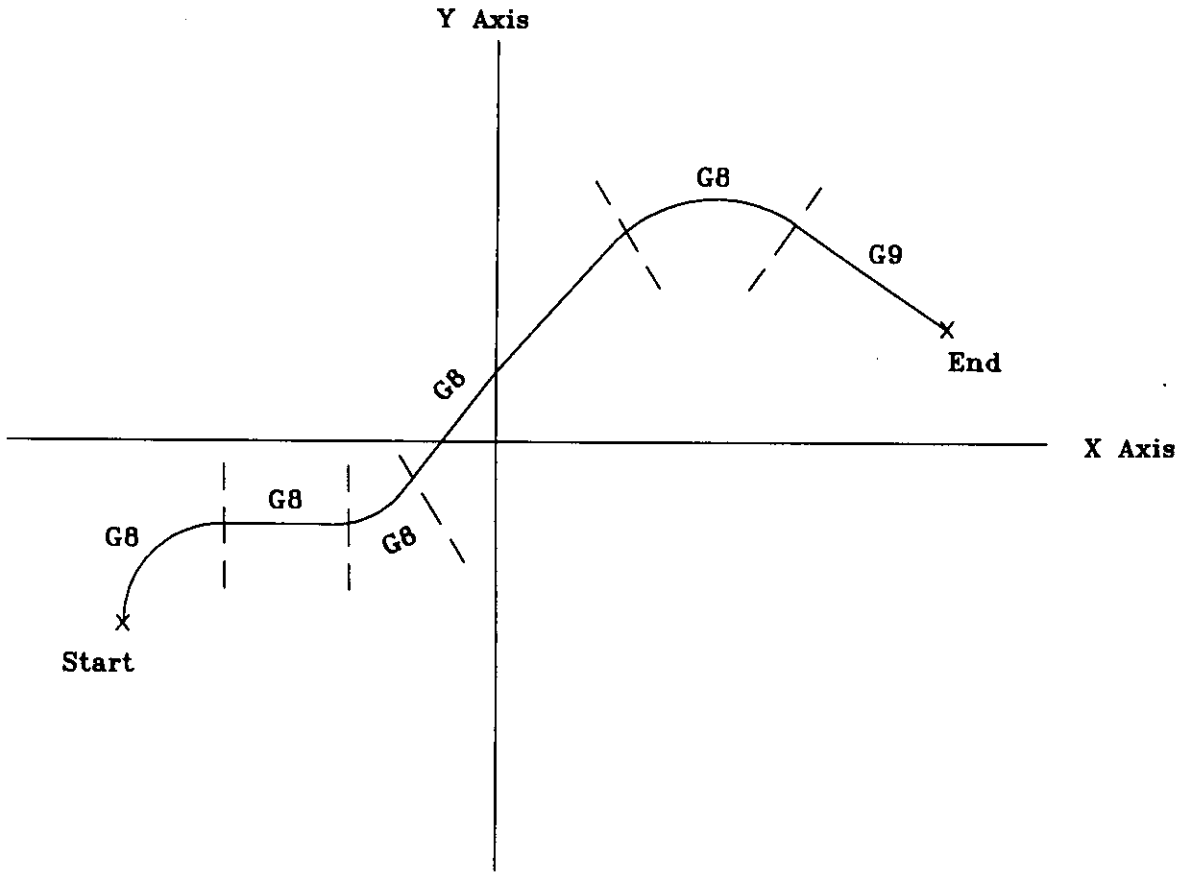


Figure A-3: Typical Two Axis Position Profile Using "G8" and "G24" Modes

APPENDIX 3: PROGRAMMING EXAMPLES

PROGRAMMING EXAMPLE #1:

This example utilizes the following commands:

$OT7	**DENT**	**G1**	**G2**
G3	**G70**	**G90**	**G91**
JUMP	**M0**	**M2**	**MSG**
SCF	**SCO**		

```
(DENT,ENT1)                                  ; Define the Entry Point ENT1.
(SCF,X.4,Y.4)                                ; Scaling factor, X.4 Y.4.
(SCO,1)                                      ; Scaling is ON.
G91 G70                                      ; Initiate Incremental Positioning and
                                             ; English Programming (inches).
(DENT,HFCL)                                  ; Define the Entry Point HFCL.
(MSG,---press cycle start to run 4 half circles---)  ; Display a message.
M0                                           ; Program Stop
$OT7=H,0                                     ; Pen down, Output 7 is ON.
G2 X2. Y0. I1. J0.                           ; 1st 1/2 circle CW
G3 X2. Y0. I1. J0.                           ; 2nd 1/2 circle CCW
G3 X-2. Y0. I-1. J0.                         ; 3rd 1/2 circle CCW
G2 X-2. Y0. I-1. J0.                         ; 4th 1/2 circle CW
$OT7=H,1                                     ; Pen up, Output 7 is OFF.
(DENT,QRCL)                                  ; Define the Entry Point QRCL.
G90 G1 X.5 Y3.5 F300.                        ; Absolute Linear Offset move.
(MSG,---press cycle start to run 8 qtr circles---)   ; Display a message.
M0                                           ; Program Stop
(SCO,0)                                      ; Scaling is turned OFF.
$OT7=H,0                                     ; Pen down, Output 7 is ON.
G2 X1. Y1. I1. J0.                           ; 1st 1/4 circle CW
G2 X1. Y-1. I0. J-1.                         ; 2nd 1/4 circle CW
G3 X1. Y-1. I1. J0.                          ; 3rd 1/4 circle CCW
G3 X1. Y1. I0. J1.                           ; 4th 1/4 circle CCW
G3 X-1. Y1. I-1. J0.                         ; 5th 1/4 circle CCW
G3 X-1. Y-1. I0. J-1.                        ; 6th 1/4 circle CCW
```

PROGRAMMING EXAMPLE #1 (CON'T):

```
G2 X-1. Y-1. I-1. J0.          ; 7th 1/4 circle CW
G2 X-1. Y1. I0. J1.            ; 8th 1/4 circle CW
$OT7=H,1                       ; Pen up, Output 7 is OFF.
(JUMP,ENT1)                    ; Jump to Entry Point ENT1.
M2                             ; End of The Program
```

PROGRAMMING EXAMPLE #2:

This example utilizes the following commands:

$OT7	**$TOD**	**DENT**	**G1**
G4	**G24**	**G70**	**G90**
G91	**JUMP**	**M0**	**M2**
MTOR	**MSG**	**RAMP**	**RPT**

```
(MSG,#$TOD)                                          ; Display the Time and
                                                     ; Date.
(DENT,STRT)                                          ; Define Entry Point STRT.
G91 G70                                              ; Initiate Incremental Posi-
                                                     ; tioning and English
                                                     ; Programming (inches).
G24                                                  ; Non-Corner Rounding
                                                     ; mode.
(DENT,SCA1)                                          ; Define Entry Point SCA1.
G90 G1 X0. Y0. F300.                                 ; ABS Linear Offset move.
G91                                                  ; Initiate Incremental Posi-
                                                     ; tioning.
(MSG,---press cycle start to continue with scan pattern- #1---)  ; Display a message.
M0                                                   ; Program Stop
(MSG,)                                               ; Clear the message.
$OT7=H,0                                             ; Pen down, Output 7 is ON.
(RAMP,250)                                           ; Ramp time is .250 sec-
                                                     ; onds.
```

PROGRAMMING EXAMPLE #2 (CON'T):

```
(RPT,10                                              ; Scan will move Y in/out 5",
                                                     ; inc. of .25" on X.
 G1 Y5. F300.                                        ; Linear Contouring move.
 G1 X.25 F100.                                       ; Linear Contouring move.
 G1 Y-5. F300.                                       ; Linear Contouring move.
 G1 X.25 F100.                                       ; Linear Contouring move.
)                                                    ; Close the Repeat Loop.
Y5. F300.                                            ; Linear Contouring move.
$OT7=H,1                                             ; Pen up, Output 7 is OFF.
(DENT,SCA2)                                          ; Define Entry Point SCA2.
G90 G1 X0. Y0. F300.                                 ; ABS Linear Offset move.
G91                                                  ; Initiate Incremental Posi-
                                                     ; tioning.
(MSG,---press cycle start to con't with scan pattern #2---)  ; Display a message.
M0                                                   ; Program Stop
(MSG,)                                               ; Clear the message.
$OT7=H,0                                             ; Pen down, Output 7 is ON.
(RAMP,100)                                           ; Ramp for 100 milliseconds
(RPT,10                                              ; Repeat Loop
 G1 X5. F300.                                        ; Linear Contouring move.
 G1 Y.25 F100.                                       ; Linear Contouring move.
 G1 X-5. F300.                                       ; Linear Contouring move.
 G1 Y.25                                             ; Linear Contouring move.
)                                                    ; Close the Repeat Loop.
G1 X5. F300.                                         ; Linear Offset move.
$OT7=H,1                                             ; Pen up, Output 7 is OFF.
(MSG,---press cycle start to return absolute X0/Y0---)  ; Display a message.
M0                                                   ; Program Stop
G90 G70                                              ; Initiate Absolute Position-
                                                     ; ing and English Program-
                                                     ; ming (inches).
G1 X0. Y0. F300.                                     ; Linear Offset move.
G91                                                  ; Initiate Incremental Posi-
                                                     ; tioning.
(MTOR,0,X,Y)                                         ; Motors OFF for X, Y axis.
```

PROGRAMMING EXAMPLE #2 (CON'T):

```
(MSG, motors OFF, move X and Y, press cycle start to con't)   ; Display a message.
M0                                                            ; Program Stop
(MSG,---load new parts in fixture-press cycle start to con't---)  ; Display a message.
M0                                                            ; Program Stop
(MSG, clear all personnel-press cycle start to power motors)  ; Display a message.
M0                                                            ; Program Stop
(MSG, ---press cycle start to repeat program---)              ; Display a message.
(MTOR,1,X,Y)                                                  ; Motors ON for X, Y axis.
G4 F1.                                                        ; A Dwell of 1 second is es-
                                                              ; tablished.
M0                                                            ; Program Stop
(MSG,)                                                        ; Clear the message.
(JUMP,STRT)                                                   ; Jump to Entry Point STRT.
M2                                                            ; End of The Program
```

PROGRAMMING EXAMPLE #3:

This example utilizes the following commands:

DENT	**G4**	**INP**	**JUMP**
M0	**M2**	**MSG**	**OTP**

```
N1 (DENT,STRT)                                  ; Define the Entry Point STRT.
N3 (MSG,#H:$INP)                                ; Display input status of 16 inputs in hex.
N4 (MSG,#H:$IN0,#H:$IN1,#H:$IN2,#H:$IN3)        ; Display value of inputs 0,1,2,and 3.
N5 (MSG,                 )                      ; Display a message.
N6 $OTP=H,00                                    ; All outputs ON (reverse logic) (powers up
                                                ; with outputs OFF).
N7 $OTP=H,FF                                    ; All outputs are OFF.
N8 $OTP=H,FE                                    ; Output 1 is ON.
N9 $OTP=H,FC                                    ; Outputs 1 and 2 are ON.
N10 $OTP=H,F8                                   ; Outputs 1,2, and 3 are ON.
N11 $OTP=H,F0                                   ; Outputs 1,2,3, and 4 are ON.
N12 $OTP=H,E0                                   ; Outputs 1,2,3,4, and 5 are ON.
N13 $OTP=H,C0                                   ; Outputs 1,2,3,4,5, and 6 are ON.
N14 $OTP=H,80                                   ; Outputs 1,2,3,4,5,6, and 7 are ON.
N15 $OTP=H,00                                   ; Outputs 1,2,3,4,5,6,7, and 8 are ON.
```

PROGRAMMING EXAMPLE #3 (CON'T):

```
N16 $OTP=H,FF                                  ; All outputs are OFF.
N17 G4 F1.                                     ; A Dwell of 1 second is established.
N18 (MSG,waiting on INT=0)                     ; Display a message.
N19 (DENT,LOOP)                                ; Define the Entry Point LOOP.
N20 (JUMP,CNT1,$IN0.EQ.H,0)                    ; Perform the specified jump.
N21 (JUMP,CNT2,$IN1.EQ.H,0)                    ; Perform the specified jump.
N22 (JUMP,CNT3,$IN2.EQ.H,0)                    ; Perform the specified jump.
N23 (JUMP,CNT4,$IN3.EQ.H,0)                    ; Perform the specified jump.
N24 (JUMP,CNT5,$IN4.EQ.H,0)                    ; Perform the specified jump.
N25 (JUMP,CNT6,$IN5.EQ.H,0)                    ; Perform the specified jump.
N26 (JUMP,CNT7,$IN6.EQ.H,0)                    ; Perform the specified jump.
N27 (JUMP,CNT8,$IN7.EQ.H,0)                    ; Perform the specified jump.
N28 (JUMP,LOOP)                                ; Jump to the Entry Point LOOP.
N29 G4 F1.                                     ; A Dwell of 1 second is established.
N30 (MSG, the inputs are negative logic also)  ; Display a message.
N31 G4 F1.                                     ; A Dwell of 1 second is established.
N32 (MSG, all outputs are off)                 ; Display a message.
N33 G4 F1.                                     ; A Dwell of 1 second is established.
N34 (DENT,CNT1)                                ; Define the Entry Point CNT1.
N35 (MSG, DENT CNT1 "H,0" INPUT 1 is on)       ; Display a message.
N36 M0                                         ; Program Stop
N37 (JUMP,STRT)                                ; Jump to Entry Point STRT.
N38 (DENT,CNT2)                                ; Define the Entry Point CNT2.
N39 (MSG, DENT CNT2 "H,0" INPUT 2 is on)       ; Display a message.
N40 M0                                         ; Program Stop
N41 (JUMP,STRT)                                ; Jump to Entry Point STRT.
N42 (DENT,CNT3)                                ; Define the Entry Point CNT3.
N43 (MSG, DENT CNT3 "H,0" INPUT3 is on)        ; Display a message.
N44 M0                                         ; Program Stop.
N45 (JUMP,STRT)                                ; Jump to Entry Point STRT.
N46 (DENT,CNT4)                                ; Define the Entry Point CNT4.
N47 (MSG, DENT CNT4 "H,0" INPUT 4 is on)       ; Display a message.
N48 M0                                         ; Program Stop
N49 (JUMP,STRT)                                ; Jump to Entry Point STRT.
N50 (DENT,CNT5)                                ; Define the Entry Point CNT5.
```

PROGRAMMING EXAMPLE #4:

This example utilizes the following commands:

CLS	**DFS**	**G4**	**INT1**
INT2	**M0**	**M2**	**M30**
M47	**MSG**		

```
(CLS,MSG1)                                              ; Call Subroutine MSG1.
(INT1,4,IN01)                                           ; Jump to DFS, IN01 upon
                                                        ; INT1 going low.
(INT2,4,IN02)                                           ; Jump to DFS, IN02 upon
                                                        ; INT2 going low.
M47                                                     ; Program Repeat
M2                                                      ; End of The Program
(DFS,MSG1                                               ; Define Subroutine MSG1.
  (MSG, testing interrupts 1 and 2)                     ; Display a message.
  G4 F.5                                                ; Dwell 1/2 sec or 500 msec
  (MSG,)                                                ; Clear the message.
  G4 F.5                                                ; Dwell 1/2 sec or 500 msec
)                                                       ; End of Subroutine.
(DFS,IN01                                               ; Define Subroutine IN01.
  (MSG, you have activated INT1, press cycle start to con't.)  ; Display a message.
  M0                                                    ; Program Stop
  (MSG,)                                                ; Clear the message.
)                                                       ; End of Subroutine IN01.
(DFS,IN02                                               ; Define Subroutine IN02.
  (MSG, you have activated INT2, press cycle start to con't.)  ; Display a message.
  M0                                                    ; Program Stop
  (MSG,)                                                ; Clear the message.
)                                                       ; End of Subroutine IN02.
M30                                                     ; Physical End of Program
```

PROGRAMMING EXAMPLE #5:

This example utilizes the following commands:

CCP	**G1**	**G40**	**G41**
G42	**G70**	**G91**	**REF**

```
G70                                  ; Initiate English Programming (inches).
G91                                  ; Initiate Incremental Positioning.
(REF,X,Y)                            ; Find the Hardware Home for the X and Y axes.
G1 F200.                             ; Linear moves with a Feedrate of 200.
(CCP,X0.,Y.2)                        ; Cutter Compensation for a tool Dia of .25".
; ***************************************Without Cutter Compensation left
X2. Y2.                              ; Linear Offset move for the X and Y axes.
Y2.                                  ; Linear move of the Y axis.
X2.                                  ; Linear move of the X axis.
Y-2.                                 ; Linear move of the Y axis.
X-2.                                 ; Linear move of the X axis.
X-2. Y-2.                            ; Move back to Home or the starting point.
; ***************************************Cutter Compensation left
G41 X2. Y2.                          ; Initiate Cutter Compensation left with index.
Y2.                                  ; Linear move of the Y axis.
X2.                                  ; Linear move of the X axis.
Y-2.                                 ; Linear move of the Y axis.
X-2.                                 ; Linear move of the X axis.
G40                                  ; Deactivate the Cutter Compensation.
X-2. Y-2.                            ; Move back to Home or the starting point.
; ***************************************Without Cutter Compensation right
X6. Y2.                              ; Linear Offset move of the X and Y axes.
X2.                                  ; Linear move of the X axis.
Y2.                                  ; Linear move of the Y axis.
X-2.                                 ; Linear move of the X axis.
Y-2.                                 ; Linear move of the Y axis.
X-6. Y-2.                            ; Move back to Home or the starting point.
; ***************************************Cutter Compensation right
G42 X6. Y2.                          ; Initiate Cutter Compensation right with index.
X2.                                  ; Linear move of the X axis.
```

PROGRAMMING EXAMPLE #5 (CON'T):

```
Y2.                    ; Linear move of the Y axis.
X-2.                   ; Linear move of the X axis.
Y-2.                   ; Linear move of the Y axis.
G40                    ; Deactivate Cutter Compensation.
X-6. Y-2.              ; Move back to Home or the starting point.
M47                    ; Restart The Program
```

PROGRAMMING EXAMPLE #6:

This example utilizes the following commands:

G1	**G23**	**G24**	**G70**
G90	**M0**	**REF**	

```
G70                    ; Initiate English Programming (inches).
G91                    ; Initiate Incremental Positioning.
(REF,X,Y)              ; Find Hardware Home for the X and Y axes.
G1 X2. Y2. F200.       ; Linear move with Feedrate of 200.
; **************************************Corner Rounding Mode
G23                    ; Corner Rounding mode
G1 X2.                 ; Linear move of the X axis.
Y2.                    ; Linear move of the Y axis.
X-2.                   ; Linear move of the X axis.
Y-2.                   ; Linear move of the Y axis.
M0                     ; Program Stop
; **************************************Non-Corner Rounding mode
G24                    ; Non-Corner Rounding mode
G1 X2.                 ; Linear move of the X axis.
Y2.                    ; Linear move of the Y axis.
X-2.                   ; Linear move of the X axis.
Y-2.                   ; Linear move of the Y axis.
M0                     ; Program Stop
M2                     ; End of The Program
```

PROGRAMMING EXAMPLE #7:

This example utilizes the following commands:

DENT	**DVAR**	**EQ**	**JUMP**
M0	**MSG**		

```
(DVAR,VAR1,VAR2,VAR3,VAR4,VAR5,
 VAR6,FDRT,XDST,YDST)                                  ; Define indicated variables.
(MSG,<VAR1,VAR2,VAR3,VAR4>,
 Please enter your name in "quotes")                   ; Input messages.
(MSG, Welcome, #C:VAR1,VAR2,VAR3,VAR4)                 ; Display a message.
(MSG,<VAR5>, Please enter your
 employee number at this time)                         ; Input messages.
(MSG,#C:VAR1,VAR2,VAR3,VAR4, Please
 enter your employee #, #VAR5, press cycle start to con't)  ; Display a message.
M0                                                     ; Program Stop
(DENT,ENT1)                                            ; Define Entry Point ENT1.
(MSG,<VAR6>,Enter 1-Home, 2-HF circle)                 ; Display a message.
(MSG,)                                                 ; Clear the message.
(JUMP,HOME,VAR6.EQ.1)                                  ; Jump if equal to..:
(JUMP,HFCL,VAR6.EQ.2)                                  ; Jump if equal to...
(JUMP,ENT1)                                            ; Jump to Entry Point ENT1.
(DENT,CONT)                                            ; Define Entry Point CONT.
(DENT,HOME)                                            ; Define Entry Point HOME.
(REF,X,Y)                                              ; Find Hardware Home for
                                                       ; the X and Y axes.
(MSG,<FDRT>,Enter X and Y axis Feedrate)               ; Input messages.
(MSG,<XDST>, Enter X Offset)                           ; Input messages.
(MSG,<YDST>, Enter Y Offset)                           ; Input messages.
G1 X=XDST Y=YDST F=FDRT                                ; Linear X/Y Variable move.
(JUMP,ENT1)                                            ; Jump to Entry Point ENT1.
(DENT,HFCL)                                            ; Define Entry Point HFCL.
(MSG, Running HFCL, press cycle start to return)       ; Display a message.
M0                                                     ; Program Stop
(MSG,)                                                 ; Clear the message.
(JUMP,ENT1)                                            ; Jump to Entry Point ENT1.
M2                                                     ; End of The Program
```

DECIMAL	BINARY	HEX	DECIMAL	BINARY	HEX	DECIMAL	BINARY	HEX	DECIMAL	BINARY	HEX
0	00000000	00	52	00110100	34	104	01101000	68	156	10011100	9C
1	00000001	01	53	00110101	35	105	01101001	69	157	10011101	9D
2	00000010	02	54	00110110	36	106	01101010	6A	158	10011110	9E
3	00000011	03	55	00110111	37	107	01101011	6B	159	10011111	9F
4	00000100	04	56	00111000	38	108	01101100	6C	160	10100000	A0
5	00000101	05	57	00111001	39	109	01101101	6D	161	10100001	A1
6	00000110	06	58	00111010	3A	110	01101110	6E	162	10100010	A2
7	00000111	07	59	00111011	3B	111	01101111	6F	163	10100011	A3
8	00001000	08	60	00111100	3C	112	01110000	70	164	10100100	A4
9	00001001	09	61	00111101	3D	113	01110001	71	165	10100101	A5
10	00001010	0A	62	00111110	3E	114	01110010	72	166	10100110	A6
11	00001011	0B	63	00111111	3F	115	01110011	73	167	10100111	A7
12	00001100	0C	64	01000000	40	116	01110100	74	168	10101000	A8
13	00001101	0D	65	01000001	41	117	01110101	75	169	10101001	A9
14	00001110	0E	66	01000010	42	118	01110110	76	170	10101010	AA
15	00001111	0F	67	01000011	43	119	01110111	77	171	10101011	AB
16	00010000	10	68	01000100	44	120	01111000	78	172	10101100	AC
17	00010001	11	69	01000101	45	121	01111001	79	173	10101101	AD
18	00010010	12	70	01000110	46	122	01111010	7A	174	10101110	AE
19	00010011	13	71	01000111	47	123	01111011	7B	175	10101111	AF
20	00010100	14	72	01001000	48	124	01111100	7C	176	10110000	B0
21	00010101	15	73	01001001	49	125	01111101	7D	177	10110001	B1
22	00010110	16	74	01001010	4A	126	01111110	7E	178	10110010	B2
23	00010111	17	75	01001011	4B	127	01111111	7F	179	10110011	B3
24	00011000	18	76	01001100	4C	128	10000000	80	180	10110100	B4
25	00011001	19	77	01001101	4D	129	10000001	81	181	10110101	B5
26	00011010	1A	78	01001110	4E	130	10000010	82	182	10110110	B6
27	00011011	1B	79	01001111	4F	131	10000011	83	183	10110111	B7
28	00011100	1C	80	01010000	50	132	10000100	84	184	10111000	B8
29	00011101	1D	81	01010001	51	133	10000101	85	185	10111001	B9
30	00011110	1E	82	01010010	52	134	10000110	86	186	10111010	BA
31	00011111	1F	83	01010011	53	135	10000111	87	187	10111011	BB
32	00100000	20	84	01010100	54	136	10001000	88	188	10111100	BC
33	00100001	21	85	01010101	55	137	10001001	89	189	10111101	BD
34	00100010	22	86	01010110	56	138	10001010	8A	190	10111110	BE
35	00100011	23	87	01010111	57	139	10001011	8B	191	10111111	BF
36	00100100	24	88	01011000	58	140	10001100	8C	192	11000000	C0
37	00100101	25	89	01011001	59	141	10001101	8D	193	11000001	C1
38	00100110	26	90	01011010	5A	142	10001110	8E	194	11000010	C2
39	00100111	27	91	01011011	5B	143	10001111	8F	195	11000011	C3
40	00101000	28	92	01011100	5C	144	10010000	90	196	11000100	C4
41	00101001	29	93	01011101	5D	145	10010001	91	197	11000101	C5
42	00101010	2A	94	01011110	5E	146	10010010	92	198	11000110	C6
43	00101011	2B	95	01011111	5F	147	10010011	93	199	11000111	C7
44	00101100	2C	96	01100000	60	148	10010100	94	200	11001000	C8
45	00101101	2D	97	01100001	61	149	10010101	95	201	11001001	C9
46	00101110	2E	98	01100010	62	150	10010110	96	202	11001010	CA
47	00101111	2F	99	01100011	63	151	10010111	97	203	11001011	CB
48	00110000	30	100	01100100	64	152	10011000	98	204	11001100	CC
49	00110001	31	101	01100101	65	153	10011001	99	205	11001101	CD
50	00110010	32	102	01100110	66	154	10011010	9A	206	11001110	CE
51	00110011	33	103	01100111	67	155	10011011	9B	207	11001111	CE

DECIMAL	BINARY	HEX
208	11010000	D0
209	11010001	D1
210	11010010	D2
211	11010011	D3
212	11010100	D4
213	11010101	D5
214	11010110	D6
215	11010111	D7
216	11011000	D8
217	11011001	D9
218	11011010	DA
219	11011011	DB
220	11011100	DC
221	11011101	DD
222	11011110	DE
223	11011111	DF
224	11100000	E0
225	11100001	E1
226	11100010	E2
227	11100011	E3
228	11100100	E4
229	11100101	E5
230	11100110	E6
231	11100111	E7
232	11101000	E8
233	11101001	E9
234	11101010	EA
235	11101011	EB
236	11101100	EC
237	11101101	ED
238	11101110	EE
239	11101111	EF
240	11110000	F0
241	11110001	F1
242	11110010	F2
243	11110011	F3
244	11110100	F4
245	11110101	F5
246	11110110	F6
247	11110111	F7
248	11111000	F8
249	11111001	F9
250	11111010	FA
251	11111011	FB
252	11111100	FC
253	11111101	FD
254	11111110	FE
255	11111111	FF

SERVICE AND REPAIR

Customer repair of the equipment is limited. Control Board(s) may be removed and replaced if necessary, however, component level repair must not be attempted.

On-site service should be performed by an experienced electronic technician, preferably one trained by Aerotech.

SHIPMENT

The procedure for shipping equipment to Aerotech, described below, pertains to warranty as well as non-warranty repairs.

1. Before returning any equipment a "Retern Authorization Number" must be obtained from Aerotech. (Be prepared to give the serial number of the equipment being returned.)

2. The equipment being returned must be encased in a proper cushioning material and enclosed in a cardboard box.

Call for a "Return Authorization Number" if it is necessary to ship any equipment to the factory.

WARNING: DAMAGE TO THE EQUIPMENT DUE TO IMPROPER PACKAGING MAY BOID WARRANTY!

AEROTECH, INC. SALES OFFICES

Aerotech Sales and Serviece Offices are listed below. For service and information, contact the office servicing your area.

WORLD HEADQUARTERS
AEROTECH, INC.
101 Zeta Drive
Pittsburgh, PA 15238

Phone (412) 963-7470
FAX (412) 963-7459
TWX (710) 795-3125

AEROTECH, LTD.
Aldermaston
Berkshire RG7 4QW, England

Phone (07356) 77274
FAX (07356) 5022
TLX 847228
Country Code (44)

AEROTECH GMBH
Neumeyerstrasse 90
8500 Nuernberg 10
West Germany

Phone (0911) 521031
FAX (0911) 521235
TLX 622474
Country Code (49)

Warranty and Field Service Policy

Aerotech, Inc. warrants its products to be free from defects caused by faulty materials or poor workmanship for a minimum period of one year from date of shipment from Aerotech. Aerotech's liability is limited to replacing, repairing or issuing credit, at its option, for any products which are returned by the original purchaser during the warranty period. Aerotech makes no warranty that its products are fit for the use or purpose to which they may be put by the buyer, where or not such use or purpose has been disclosed to Aerotech in specifications or drawings previously or subsequently provided, or whether or not Aerotech's products are specifically designed and/or manufactured for buyer's use or purpose. Aerotech's liability on any claim for loss or damage arising out of the sale, resale or use of any of its products shall in no event exceed the selling price of the unit.

Laser Product Warranty

Aerotech, Inc. warrants its laser products to the original purchaser for a minimum period of one year from date of shipment. This warranty covers defects in workmanship and material and is voided for all laser power supplies, plasma tubes and laser systems subject to electrical or physical abuse, tampering (such as opening the housing or removal of the serial tag) or improper operation as determined by Aerotech. This warranty is also voided for failure to comply with Aerotech's return procedures.

Return Products Procedure

Claims for shipment damage (evident or concealed) must be filed with the carrier by the buyer. Aerotech must be notified within (30) days of shipment of incorrect materials. No product may be returned, whether in warranty or out of warranty, without first obtaining approval from Aerotech. No credit will be given nor repairs made for products returned without such approval. Any returned product(s) must be accompanied by a return authorization number. The return authorization number may be obtained by calling an Aerotech service center. Products must be returned, prepaid, to an Aerotech service center (no C.O.D. or Collect Freight accepted). The status of any product returned later than (30) days after the issuance of a return authorization number will be subject to review.

Returned Product Warranty Determination

After Aerotech's examination, warranty or out-of-warranty status will be determined. If upon Aerotech's examination a warrantied defect exists, then the product(s) will be repaired at no charge and shipped, prepaid, back to the buyer. If the buyer desires an air freight return, the product(s) will be shipped collect. Warranty repairs do no extend the original warranty period.

Returned Product Non-Warranty Determination

After Aerotech's examination, the buyer shall be notified of the repair cost. At such time the buyer must issue a valid purchase order to cover the cost of the repair and freight, or authorize the product(s) to be shipped back as is, at the buyer's expense. Failure to obtain a purchase order number or approval within (30) days of notification will result in the product(s) being returned as is, at the buyer's expense. Repair work is warranted for (90) days from date of shipment. Replacement components are warranted for one year from date of shipment.

Rush Service

At times, the buyer may desire to expedite a repair. Regardless of warranty or out-of-warranty status, the buyer must issue a valid purchase order to cover the added rush service cost. Rush service is subject to Aerotech's approval.

On-Site Warranty Repair

If an Aerotech product cannot be made functional by telephone assistance or by sending and having the customer install replacement parts, and cannot be returned to the Aerotech service center for repair, and if Aerotech determines the problem could be warranty-related, then the following policy applies:

Aerotech will provide an on-site field service representative in a reasonable amount of time, provided that the customer issues a valid purchase order to Aerotech covering all transportation and subsistence costs. For warranty field repairs, the customer will not be charged for the cost of labor and material. If service is rendered at times other than normal work periods, then special service rates apply.

If during the on-site repair it is determined the problem is not warranty related, then the terms and conditions stated in the following "On-Site Non-Warranty Repair" section apply.

On-Site Non-Warranty Repair

If any Aerotech product cannot be made functional by telephone assistance or purchased replacement parts, and cannot be returned to the Aerotech service center for repair, then the following field service policy applies:

Aerotech will provide an on-site field service representative in a reasonable amount of time, provided that the customer issues a valid purchase order to Aerotech covering all transportation and subsistence costs and the prevailing labor cost, including travel time, necessary to complete the repair.

AEROTECH, Inc., 101 Zeta Drive, Pittsburgh, Pennsylvania 15238

Phone (412) 963-7470 • TWX (710) 795-3125 • FAX (412) 963-7459

INDEX