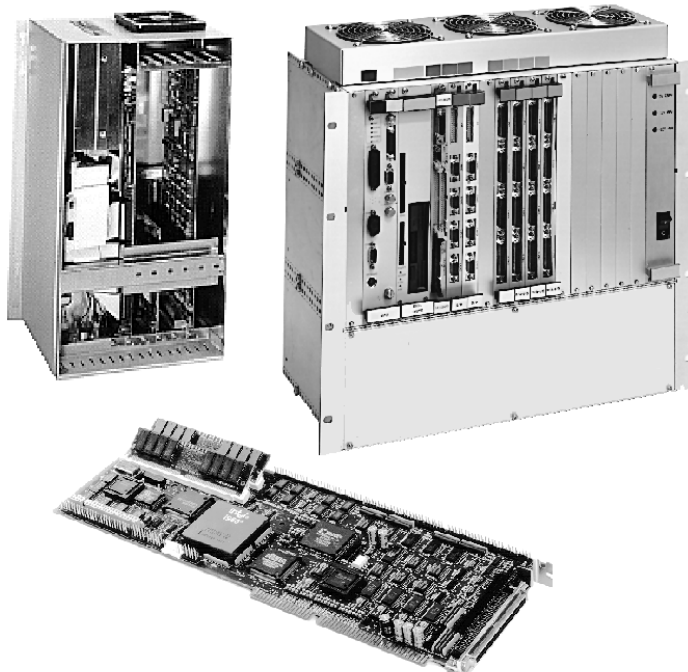




UNIDEX 600 Series PROGRAMMING MANUAL

P/N: EDU152



**AEROTECH, Inc.
101 Zeta Drive
Pittsburgh, PA. 15238-2897 U.S.A.
(412) 963-7470
Fax (412) 963-7459**

UNIDEX 631 and UNIDEX 600 are products of Aerotech, Inc.
Operating System 2 (OS/2) is a registered trademark of International Business Machines, Inc.

The UNIDEX 600 Series Programming Manual Revision History:

Rev 1.0	May, 1995
Rev 1.0a	August, 1995
Rev 1.1	June 5, 1996
Rev 1.2	December 23, 1996

TABLE OF CONTENTS

CHAPTER 1: SYMBOLS & AXIS DESIGNATORS 1-1

- 1.1. Special Symbols 1-1
 - 1.1.1. Comment Operator ; 1-1
 - 1.1.2. Block Delete Character / 1-1
 - 1.1.3. Start Extended Command Block (..... 1-1
 - 1.1.4. End Extended Command Block) 1-2
 - 1.1.5. Array Indices [] 1-2
- 1.2. Special Characters 1-3
 - 1.2.1. Line Numbers Nxxxx 1-3
 - 1.2.2. Linear Feedrate F 1-4
 - 1.2.3. Feedrate Limiting 1-5
 - 1.2.3.1. Notes on the Feedrates Display Screen 1-6
 - 1.2.4. Rotary Feedrate E 1-7
 - 1.2.5. Spindle Feedrate S 1-7
 - 1.2.6. Constant Surface Speed Feedrate SF 1-8
 - 1.2.7. Circle CenterPoints I/J/K 1-8
- 1.3. Parameter Types 1-8
 - 1.3.1. Floating Point Constants 1-8
 - 1.3.2. Character Strings 1-9
 - 1.3.3. Filenames 1-9
 - 1.3.4. Hexadecimal Integers 1-9
- 1.4. Variables 1-9
 - 1.4.1. Local Variables 1-10
 - 1.4.2. Global Variables 1-10
 - 1.4.3. Using the Global Alias File 1-11
 - 1.4.4. Static Variables STATICxx 1-12
 - 1.4.5. System Variables 1-13
 - 1.4.5.1. CNC Number CNCNUM 1-13
 - 1.4.5.2. CNC Time CNCTIME 1-13
 - 1.4.5.3. CNC Position Command POSITNxx 1-13
 - 1.4.5.4. CNC Preset Command PRESETnn 1-13
 - 1.4.5.5. CNC Status Words CNCSTATx 1-14
 - 1.4.5.6. CNC Operational Mode 1-17
 - 1.4.6. CNC Analog Feedback Commands 1-18
- 1.5. Tool Word Txxxx 1-19
 - 1.5.1. UNIDEX 631/U600 CNC Tool Handling 1-19
 - 1.5.1.1. Inspection Station Probe Set Data 1-21
- 1.6. Multiple CNCs 1-22
 - 1.6.1. CNC Intercommunication - Global Variables 1-22
 - 1.6.2. CNC Intercommunication - CNC Parameters Setup 1-22
 - 1.6.3. Autorun Mode (Autorun.ini) 1-22

CHAPTER 2: G-CODES.....	2-1
2.1. Description	2-1
2.2. Introduction to G-code Motion.....	2-5
2.3. Motion G-codes	2-7
2.3.1. Point-to-point Positioning at a Rapid Feedrate (Motion) G0	2-7
2.3.2. Linear Interpolation (Motion) G1.....	2-8
2.3.3. Circular Interpolation CW (Motion) G2.....	2-8
2.3.4. Circular Interpolation CCW (Motion) G3	2-9
2.3.5. Dwell G4	2-10
2.4. G8 and G9 Overview	2-11
2.4.1. Instantaneous Acceleration G8.....	2-12
2.4.2. Force Deceleration G9	2-14
2.4.3. Circular Interpolation CW (Motion) G12.....	2-16
2.4.4. Circular Interpolation CCW on Axis Plane 2 G13	2-16
2.4.5. Spline Move G30	2-17
2.4.6. Constant Lead Thread Cutting G33.....	2-18
2.5. Plane Selection Codes	2-21
2.5.1. Plane Selection Codes Set # 1 G17/G18/G19	2-21
2.6. Normalcy Mode (G20, G21, G22).....	2-22
2.6.1. Disable Normalcy Mode G20.....	2-24
2.6.2. Activate Normalcy Mode Left G21	2-24
2.6.3. Activate Normalcy Mode Right G22.....	2-25
2.7. Safe Zones (G36, G37).....	2-26
2.7.1. Enable Safe Zones G36.....	2-26
2.7.2. Disable Safe Zones G37	2-27
2.8. Intersectional Cutter Radius Compensation (ICRC) Overview	2-28
2.8.1. Deactivate Cutter Compensation (ICRC) G40	2-30
2.8.2. Activate ICRC Right G41.....	2-31
2.8.3. Activate ICRC Left G42.....	2-33
2.8.4. Set Cutter Compensation Radius G43	2-35
2.8.5. Set Cutter Compensation Axes G44	2-35
2.8.6. Disable Polar or Cylindrical Coordinate Transformation G45	2-36
2.8.7. Enable Polar Coordinate Transformation G46	2-36
2.8.8. Enable Cylindrical Coordinate Transformation G47	2-37
2.9. Fixture Offset (G53, G54, G55)	2-39
2.9.1. Cancel Fixture Offset G53.....	2-39
2.9.2. Set Fixture Offset #1 G54.....	2-39
2.9.3. Set Fixture Offset #2 G55.....	2-40
2.10. Parameter Monitoring (G56, G57)	2-41
2.10.1. Enable Parameter Monitoring G56.....	2-41
2.10.2. Disable Parameter Monitoring G57.....	2-41
2.11. Acceleration/Deceleration Overview (G60, G61).....	2-42
2.11.1. Set Acceleration Time G60	2-43
2.11.2. Set Deceleration Time G61	2-44
2.11.3. Set Profile Time G62.....	2-45
2.11.4. Sinusoidal (1-Cosine) Acceleration Mode G63.....	2-46
2.11.5. Linear Acceleration Mode G64.....	2-46
2.11.6. Set Acceleration Rate G65	2-47

- 2.11.7. Set Deceleration Rate G66 2-48
- 2.11.8. Acceleration/Deceleration Time Based (Ramp Type)
G67..... 2-49
- 2.11.9. Acceleration/Deceleration Rate Based (Ramp Type)
G68..... 2-49
- 2.12. Programming Codes 2-50
 - 2.12.1. Inch Dimension Programming Mode (Units) G70 2-50
 - 2.12.2. Metric Dimension Programming Mode (Units)
G71..... 2-51
 - 2.12.3. Restore Position Registers G82..... 2-51
 - 2.12.4. Mirror Image G83 2-52
 - 2.12.5. Parts Rotation G84 2-54
 - 2.12.6. Absolute Dimension Programming Mode (Distance)
G90..... 2-56
 - 2.12.7. Incremental Position Programming (Distance) G91..... 2-57
 - 2.12.8. Software Home (Set Position Registers) G92 2-58
- 2.13. Feedrate and Spindle Speed Codes..... 2-59
 - 2.13.1. Inverse Time Feedrate Programming (FeedrateMode)
G93..... 2-59
 - 2.13.2. Feed Per Minute Feedrate Programming
(FeedrateMode) G94 2-60
 - 2.13.3. Feed Per Spindle Revolution Feedrate Programming
G95..... 2-61
 - 2.13.4. Constant Surface Speed Spindle Programming
G96..... 2-63
 - 2.13.5. Direct RPM Spindle Programming G97..... 2-64
- 2.14. Dominant Feedrate Overview (G98, G99)..... 2-65
 - 2.14.1. Rotary Feedrate Dominant G98..... 2-67
 - 2.14.2. Linear Feedrate Dominant G99 2-68
- 2.15. Circular Direction Codes 2-69
 - 2.15.1. Normal Circular Interpolation G110 2-69
 - 2.15.2. Inverse Circular Interpolation G111..... 2-70
 - 2.15.3. 4 Kilo-Hertz Servo Update Rate G130..... 2-71
 - 2.15.4. 1 Kilo-Hertz Servo Update Rate G131..... 2-71

CHAPTER 3: M-CODES3-1

- 3.1. Description 3-1
 - 3.1.1. Program Stop M0 3-3
 - 3.1.2. Optional Stop M01 3-3
 - 3.1.3. End Of Program M02..... 3-3
 - 3.1.4. Spindle On Clockwise M03 3-3
 - 3.1.5. Spindle On Counterclockwise M04..... 3-3
 - 3.1.6. Spindle Off M05 3-3
 - 3.1.7. Spindle Off/Reorient M19..... 3-3
 - 3.1.8. Restart Program Execution and Wait for Cycle Start
M30 3-4
 - 3.1.9. Restart Program Execution M47 3-4
 - 3.1.10. Feedrate Override Lock M48 3-4
 - 3.1.11. Feedrate Override Unlock M49..... 3-4
 - 3.1.12. Spindle Feedrate Override Lock M50..... 3-4
 - 3.1.13. Spindle Feedrate Override Unlock M51 3-4

3.2.	M-code Programming	3-5
3.3.	Assigning Virtual I/O.....	3-5
3.3.1.	The MODSCAN.INI File	3-8
3.3.2.	Programmable Logic Controllers PLC	3-9
3.3.3.	Associating Virtual I/O with PLC I/O	3-10
3.3.4.	PCDIO Digital I/O Cards	3-11
3.3.4.1.	Define PCDIO I/O Board.....	3-11
3.3.5.	XYCOM Digital I/O Cards XYCOM.....	3-13
3.3.6.	Define XYCOM I/O Board	3-13
3.3.7.	Associating Virtual I/O with Xycom I/O.....	3-14
3.3.8.	The MCODEx.INI File.....	3-15
3.3.9.	Binary Input M-code Initialization	3-18
3.3.10.	Binary Output M-code Initialization	3-20
3.3.11.	Register Input M-code Initialization.....	3-22
3.3.12.	Register Output M-code Initialization	3-23
3.4.	Automatically Initializing Virtual I/O.....	3-25
3.5.	The FAULTMSG.INI File.....	3-25

CHAPTER 4: EXTENDED COMMANDS4-1

4.1.	Description	4-1
4.2.	Defining Commands	4-5
4.2.1.	Define Symbolic Constant DEFINE.....	4-5
4.2.2.	Define Local Variable DVAR	4-6
4.2.3.	Define Local Variable Arrays DVAR	4-7
4.2.4.	Define Entry Point DENT	4-8
4.2.5.	Define Subroutine DFS	4-8
4.3.	Programming Operators.....	4-10
4.3.1.	Arithmetic Operators.....	4-10
4.3.1.1.	Addition +	4-10
4.3.1.2.	Subtraction -.....	4-10
4.3.1.3.	Multiplication *.....	4-10
4.3.1.4.	Division /.....	4-10
4.3.1.5.	Modulus MOD	4-10
4.3.1.6.	Exponentiation ^	4-11
4.3.1.7.	Absolute Value ABS.....	4-11
4.3.1.8.	Truncation INT	4-11
4.3.1.9.	Fractional FRAC	4-11
4.3.1.10.	Square Root SQRT	4-11
4.3.1.11.	Precedence ().....	4-12
4.3.2.	Trigonometric Operators	4-13
4.3.2.1.	Sine SIN.....	4-13
4.3.2.2.	Cosine COS.....	4-13
4.3.2.3.	Tangent TAN	4-13
4.3.2.4.	Arcsine ASIN.....	4-13
4.3.2.5.	Arccosine ACOS.....	4-13
4.3.2.6.	Arctangent ATAN.....	4-13
4.3.3.	Logical Operators.....	4-14
4.3.3.1.	Negation NOT.....	4-14
4.3.3.2.	And AND.....	4-14
4.3.3.3.	Or OR.....	4-14
4.3.3.4.	Exclusive Or XOR	4-14

	4.3.3.5. Shift Left SHL.....	4-15
	4.3.3.6. Shift Right SHR	4-15
4.3.4.	Relational Operators.....	4-15
	4.3.4.1. Equal To EQ	4-15
	4.3.4.2. Not Equal To NE	4-15
	4.3.4.3. Greater Than GT.....	4-16
	4.3.4.4. Greater Than or Equal To GE.....	4-16
	4.3.4.5. Less Than LT	4-16
	4.3.4.6. Less Than or Equal To LE.....	4-16
4.4.	Commands which Affect Program Flow.....	4-17
	4.4.1. Jump to a User Defined Entry Block JUMP.....	4-18
	4.4.2. Conditional Statement IF-THEN-ELSE-ENDIF	4-19
	4.4.3. Repeat Loop RPT/ENDRPT	4-21
	4.4.4. Conditional Branch On Error Conditions ONERRGOTO	4-22
	4.4.5. Conditional Looping WHILE-DO-ENDWHILE	4-24
	4.4.6. Call Subroutine/Call Library Subroutine CLS/CLLS	4-26
	4.4.7. Execute OS/2 or DOS Program EXECUTE.....	4-27
4.5.	Custom Display Window Commands.....	4-28
	4.5.1. Open Custom Display Window OPENCDW	4-28
	4.5.2. Close Custom Display Window CLOSECDW	4-29
	4.5.3. Placing Items in the Window DISPLAY.....	4-29
	4.5.4. Activate CDW Log File RECORDON.....	4-32
	4.5.5. Deactivate CDW Log File RECORDOFF.....	4-33
4.6.	Synchronous Motion Commands.....	4-34
	4.6.1. ENDM Command ENDM.....	4-34
	4.6.2. Free FREE.....	4-34
	4.6.3. Handwheel Command HAND.....	4-34
	4.6.4. Home HOME/REF.....	4-35
	4.6.5. Slew Command SLEW.....	4-36
4.7.	User Stack Operations	4-37
	4.7.1. Putting Data Onto the User's Stack PUSH	4-37
	4.7.2. Removing Data From the User's Stack POP.....	4-38
4.8.	Miscellaneous Extended Commands	4-41
	4.8.1. The Autofocus Command AFCO	4-41
	4.8.2. Axis Naming HARDNAMES/SOFTNAMES	4-41
	4.8.3. Joining Parts Programs INCLUDE	4-43
	4.8.4. Data Collection Control DATA	4-44
	4.8.5. Reading Axis Parameters GETPARAM	4-48
	4.8.6. Initialize Touch Probe G51	4-49
	4.8.7. Modifying Axis Parameters SETPARAM	4-50
	4.8.8. Monitor Axis Speed MONSPD.....	4-50
	4.8.9. Wait Statement WAIT.....	4-51
	4.8.10. The WTCH Statement WTCH	4-52
4.9.	File Operation Commands.....	4-53
	4.9.1. File Open Command FILEOPEN.....	4-53
	4.9.2. FILEEOF Command FEOF.....	4-53
	4.9.3. File Read Command FILEREAD.....	4-54
	4.9.4. File Close Command FILECLOSE	4-54
	4.9.5. File Reset Command FILERESET.....	4-54
	4.9.6. File Write Command FILEWRITE.....	4-55

4.10.	Master-slave Motion Commands	4-56
4.10.1.	CONFIGM Command CONFIGM	4-57
4.10.2.	SYNC Command SYNC	4-57
4.10.3.	FEDM Command FEDM	4-58
4.11.	Asynchronous Motion Commands.....	4-59
4.11.1.	FEDM Command FEDM	4-59
4.11.2.	Index Statement INDEX.....	4-59
4.11.3.	Oscillate Command OSC.....	4-60
4.11.4.	MOVETO Statement MOVETO.....	4-60
4.11.5.	STRM Command	4-61

CHAPTER 5: OPTIONAL PSO COMMANDS5-1

5.1.	Introduction	5-1
5.2.	Programming Commands	5-2
5.3.	Conditional Tracking Based on Input States PSOC.....	5-3
5.3.1.	MODE Arguments For PSOC	5-3
5.3.1.1.	Mode Argument - 0.....	5-3
5.3.1.2.	Mode Argument - 1.....	5-3
5.3.1.3.	Mode Argument - 2.....	5-3
5.3.1.4.	Mode Argument - 3.....	5-3
5.3.2.	PSOC Arguments	5-4
5.3.2.1.	<i>i</i> Argument	5-4
5.3.2.2.	<i>n</i> Argument	5-4
5.3.2.3.	<i>in_map</i> Argument.....	5-4
5.3.2.4.	<i>out_map</i> Argument.....	5-4
5.3.2.5.	<i>input</i> Argument	5-4
5.3.2.6.	<i>output</i> Argument	5-5
5.4.	Position Synchronized Output Firing Distance Entry PSOD.....	5-6
5.4.1.	MODE Arguments For PSOD.....	5-6
5.4.1.1.	Mode Argument -0.....	5-6
5.4.1.2.	Mode Argument - 1	5-6
5.4.1.3.	Mode Argument - 2.....	5-6
5.4.2.	PSOD Arguments	5-7
5.4.2.1.	<i>distance</i> Argument	5-7
5.4.2.2.	<i>array[x]</i> Argument.....	5-7
5.4.2.3.	<i>±n</i> Argument	5-7
5.5.	Enable/Disable Position Synchronized Output Firing PSOF.....	5-9
5.5.1.	MODE Arguments For PSOF.....	5-9
5.5.1.1.	Mode Argument - 0.....	5-9
5.5.1.2.	Mode Argument - 1	5-9
5.5.1.3.	Mode argument - 2.....	5-9
5.5.1.4.	Mode Argument - 3.....	5-9
5.5.1.5.	Mode Argument - 4.....	5-10
5.5.1.6.	Mode Argument - 5.....	5-10
5.5.2.	PSOF Arguments.....	5-10
5.5.2.1.	<i>num</i> Argument.....	5-10
5.5.2.2.	<i>axis#</i> Argument	5-11
5.5.2.3.	<i>±dist</i> Argument.....	5-11
5.6.	Position Synchronized Output Using Bit Mapping PSOM.....	5-12
5.6.1.	<i>array[x]</i> Argument.....	5-12

- 5.6.2. *±size* Argument..... 5-12
- 5.7. Position Synchronized Output Pulse Configuration PSOP 5-14
 - 5.7.1. MODE Arguments For PSOP 5-14
 - 5.7.1.1. Mode Argument - 0..... 5-14
 - 5.7.1.2. Mode Argument - 1 5-14
 - 5.7.1.3. Mode Argument - 2..... 5-14
 - 5.7.1.4. Mode Argument - 3..... 5-14
 - 5.7.1.5. Mode Argument - 4..... 5-15
 - 5.7.1.6. Mode Argument - 5..... 5-15
 - 5.7.2. PSOP Arguments..... 5-15
 - 5.7.2.1. *l* Argument 5-15
 - 5.7.2.2. *w* Argument..... 5-15
 - 5.7.2.3. *t* Argument 5-15
 - 5.7.2.4. *r* Argument..... 5-15
 - 5.7.2.5. *g* Argument 5-16
 - 5.7.2.6. *array[x]* Argument 5-16
 - 5.7.2.7. *±m* Argument 5-16
- 5.8. Position Synchronized Output with Real-time Control PSOR..... 5-17
 - 5.8.1. MODE Arguments For PSOR..... 5-17
 - 5.8.1.1. Mode Argument - 0..... 5-17
 - 5.8.1.2. Mode Argument - 1 5-17
 - 5.8.1.3. Mode Argument - 3..... 5-17
- 5.9. Digital/Analog Output Command PSOT 5-18
 - 5.9.1. MODE Arguments For PSOT 5-18
 - 5.9.1.1. Mode Argument - 0..... 5-18
 - 5.9.1.2. Mode Argument - 1 5-18
 - 5.9.1.3. Mode Argument - 2..... 5-18
 - 5.9.1.4. Mode Argument - 4..... 5-18
 - 5.9.1.5. Mode Argument - 6..... 5-19
 - 5.9.1.6. Mode Argument - 8..... 5-19
 - 5.9.1.7. *on_time#* Argument 5-19
 - 5.9.1.8. *off_time#* Argument 5-19
 - 5.9.1.9. *min_off_time#* Argument 5-19
 - 5.9.1.10. *vel#* Argument..... 5-19
 - 5.9.2. PSOT Arguments 5-20
 - 5.9.2.1. *bit#* Argument 5-20
 - 5.9.2.2. *state* Argument 5-20
 - 5.9.2.3. *states* Argument 5-20
 - 5.9.2.4. *dac#* Argument..... 5-20
 - 5.9.2.5. *voltage* Argument..... 5-20
 - 5.9.2.6. *v0* Argument..... 5-20
 - 5.9.2.7. *vmax* Argument 5-21
 - 5.9.2.8. *velocity* Argument 5-21
 - 5.9.2.9. *position* Argument..... 5-21

APPENDIX A: AXIS PARAMETERS..... A-1
Description..... A-1

APPENDIX B: WARRANTY AND FIELD SERVICE..... B-1

APPENDIX C: ERROR CODES..... C-1
Description.....C-1
Axis Processor Error Codes.....C-31

INDEX

▽ ▽ ▽

LIST OF FIGURES

Figure 1-1. Renaming Variables 1-11
 Figure 1-2. The Autorun.ini File 1-23

Figure 2-1. CW Circular Interpolation 2-9
 Figure 2-2. CCW Circular Interpolation 2-10
 Figure 2-3. G8 and G9 Velocity Profile 2-11
 Figure 2-4. Velocity Profile With G8 2-12
 Figure 2-5. Velocity Profile without G9 2-14
 Figure 2-6. Velocity Profile with G9 2-15
 Figure 2-7. Effect of Splining Algorithm 2-17
 Figure 2-8. Threading Axes Designations 2-18
 Figure 2-9. Thread Taper Angle 2-19
 Figure 2-10. Threading Start Angle 2-19
 Figure 2-11. Tool Orientation 2-22
 Figure 2-12. Normalcy Left 2-24
 Figure 2-13. Normalcy Right 2-25
 Figure 2-14. Unrestricted Safe Zones 2-27
 Figure 2-15. Cutter Radius Compensation Path 2-28
 Figure 2-16. Cutter Compensation with Intervening Statements 2-29
 Figure 2-17. Lead Off Moves 2-30
 Figure 2-18. Path Compensation Right 2-31
 Figure 2-19. Lead-on Moves 2-32
 Figure 2-20. Path Compensation Right 2-33
 Figure 2-21. Effect of Cutter Compensation on Tool Path 2-34
 Figure 2-22. X, Y, Rotational and Optional Infeed Axis 2-37
 Figure 2-23. Constant Acceleration vs. 1-Cosine 2-42
 Figure 2-24. G83 Mirror Image Example 2-53
 Figure 2-25. G84 Parts Rotation Example 2-55
 Figure 2-26. Absolute Mode Programming 2-56
 Figure 2-27. Incremental Mode Programming 2-57

Figure 3-1. MODSCAN.INI File 3-8
 Figure 3-2. MCODEx.INI File 3-17
 Figure 3-3. VIRT.INI File 3-25

Figure 4-1. U600 User Interrupt 4-24
 Figure 4-2. Push/Pop Example 4-38
 Figure 4-3. Push Global Example 4-39
 Figure 4-4. Push Local Example 4-40
 Figure 4-5. Master/Slave Profile 4-56

Figure 5-1. Bit Mapping Scan Patterns for PSOM Commands 5-13

Figure A-1. The Defined Slope of a Velocity Curve at a Specified Angle A-14



LIST OF TABLES

Table 1-1. Hard Axis Names..... 1-3
Table 1-2. CNCSTAT1, CNCSTAT2, and CNCSTAT3 Variables..... 1-15
Table 1-3. CNCMODE1 and CNCMODE2 Variables 1-17

Table 2-1. G-code Summary Con't..... 2-3
Table 2-2. Axis Planes for G-codes in Plane Select Group..... 2-21
Table 2-3. Decoding a Block of a Selected Axis Plane..... 2-21
Table 2-4. G-Codes to Change Axes Used for Circular Interpolation..... 2-69
Table 2-5. Relationship of Arc Direction, Plane, & Circle Centerpoint..... 2-70

Table 3-1. M-code Summary 3-2
Table 3-2. Virtual I/O Inputs and Outputs 3-7

Table 4-1. Extended Command Summary 4-1
Table 4-2. Precedence of expressions 4-12
Table 4-3. Hard Axis Names..... 4-42

Table 5-1. Conventions for this section..... 5-1
Table 5-2. PSO Programming Commands Summary..... 5-2
Table 5-3. Distance Calculations for Multiple Axis Using the PSOD Command 5-8



PREFACE

This section gives you an overview of topics covered in each of the sections of this manual as well as conventions used in this manual. This manual contains information on the following topics:

CHAPTER 1: SYMBOLS & AXIS DESIGNATORS

Chapter 1 contains information on symbols and axis designators. This information relates to the identification and use of comment operators utilized by the UNIDEX 600 series.

CHAPTER 2: G-CODES

Chapter 2 supplies information that defines the various G-codes supported by the UNIDEX 631/U600 CNC and describes their specific mode of operation. The G-codes are divided into several different functional groups.

CHAPTER 3: M-CODES

Chapter 3 covers information describing the pre-defined M-codes for UNIDEX 631/U600 and their use in performing miscellaneous I/O functions from within the parts program.

CHAPTER 4: EXTENDED COMMANDS

Chapter 4 contains information that discusses a set of commands that allow the user to control program flow and perform other miscellaneous functions along with the G-codes and M-codes.

CHAPTER 5: OPTIONAL PSO COMMANDS

Chapter 5 contains information discussing optional Position Synchronized Output (PSO) commands that may be used to synchronize control with motion; most commonly used with laser firing. Detailed explanations of the functions of each PSO command are given in Chapter 5.

APPENDIX A: AXIS PARAMETERS

Appendix A briefly describes the function and use of the axis parameters supported by the UNIDEX 631/U600 motion controller.

APPENDIX B: WARRANTY AND FIELD SERVICE

Appendix B contains the warranty and field service policy for Aerotech products.

APPENDIX C: ERROR CODES

Appendix C provides a ready reference of Mainmenu.exe error codes listed in alphabetical order, also provided is a list of error codes for the axis processor board.

INDEX

The index contains a page number reference of topics discussed in this manual. Locator page references in the index contain the chapter number (or appendix letter) followed by the page number of the reference. Locator page numbers appear in one style: standard serif font (e.g., 3-1).

CUSTOMER SURVEY FORM

A customer survey form is included at the end of this manual for the reader's comments and suggestions about this manual. Reader's are encouraged to critique the manual and offer their feedback by completing the form and either mailing or faxing it to Aerotech.

Throughout this manual the following conventions are used:



- The terms UNIDEX 631/U600 and U631/U600 are used interchangeably throughout this manual
- The modal symbol (see left) appears in the outer margin next to commands that are modal for quick reference
- *Italic font* is used to illustrate syntax and arguments for G-codes, M-codes, and programming commands
- Note symbols (see left) appear in the outer margins next to notes following sections or paragraphs
- Anywhere in the text where there is a "xxxx", this represents the use of numbers specified by the user depending upon the application (e.g., Nxxxx specifies a parts program line number "N101", Txxxx specifies the number of the cutting tool that is being used)
- MFO is an acronym for Manual Feed Override
- MSO is an acronym for Manual Speed Override
- CNC is an acronym for Computerized Numerical Controller
- MDI is an acronym for Manual Data Interface
- This manual uses the symbol "∇ ∇ ∇" to indicate the end of a chapter.

Program lines (or blocks) are strings of characters terminated by a carriage return or line feed, or both.

Program lines consist of program words, each separated by white space. This white space is a string of any number of spaces, commas or tabs. For example, the following lines are reasonable equivalents.

```
G1,, X7
G1 X7
```

In many cases white space is not needed (for example, G1X7 is OK), but it is recommended because the CNC may not be able to decipher the words properly without white space.

The programming commands presented in this manual are grouped according to their mode of operation. Most commands appear on an individual page with an example of their usage following it.

G-codes, M-codes, and extended commands appear in **bold face** letters within their respective chapters.

All reserved words (i.e., axis names, commands, etc.) may not be used as variables, labels or soft axis names.

Although every effort has been made to ensure consistency, subtle differences may exist between the illustrations in this manual and the component and/or software screens that they represent.



PREFACE

This section gives you an overview of topics covered in each of the sections of this manual as well as conventions used in this manual. This manual contains information on the following topics:

CHAPTER 1: SYMBOLS & AXIS DESIGNATORS

Chapter 1 contains information on symbols and axis designators. This information relates to the identification and use of comment operators utilized by the UNIDEX 600 series.

CHAPTER 2: G-CODES

Chapter 2 supplies information that defines the various G-codes supported by the UNIDEX 631/U600 CNC and describes their specific mode of operation. The G-codes are divided into several different functional groups.

CHAPTER 3: M-CODES

Chapter 3 covers information describing the pre-defined M-codes for UNIDEX 631/U600 and their use in performing miscellaneous I/O functions from within the parts program.

CHAPTER 4: EXTENDED COMMANDS

Chapter 4 contains information that discusses a set of commands that allow the user to control program flow and perform other miscellaneous functions along with the G-codes and M-codes.

CHAPTER 5: OPTIONAL PSO COMMANDS

Chapter 5 contains information discussing optional Position Synchronized Output (PSO) commands that may be used to synchronize control with motion; most commonly used with laser firing. Detailed explanations of the functions of each PSO command are given in Chapter 5.

APPENDIX A: AXIS PARAMETERS

Appendix A briefly describes the function and use of the axis parameters supported by the UNIDEX 631/U600 motion controller.

APPENDIX B: WARRANTY AND FIELD SERVICE

Appendix B contains the warranty and field service policy for Aerotech products.

APPENDIX C: ERROR CODES

Appendix C provides a ready reference of Mainmenu.exe error codes listed in alphabetical order, also provided is a list of error codes for the axis processor board.

INDEX

The index contains a page number reference of topics discussed in this manual. Locator page references in the index contain the chapter number (or appendix letter) followed by the page number of the reference. Locator page numbers appear in one style: standard serif font (e.g., 3-1).

CUSTOMER SURVEY FORM

A customer survey form is included at the end of this manual for the reader's comments and suggestions about this manual. Reader's are encouraged to critique the manual and offer their feedback by completing the form and either mailing or faxing it to Aerotech.

Throughout this manual the following conventions are used:



- The terms UNIDEX 631/U600 and U631/U600 are used interchangeably throughout this manual
- The modal symbol (see left) appears in the outer margin next to commands that are modal for quick reference
- *Italic font* is used to illustrate syntax and arguments for G-codes, M-codes, and programming commands
- Note symbols (see left) appear in the outer margins next to notes following sections or paragraphs
- Anywhere in the text where there is a "xxxx", this represents the use of numbers specified by the user depending upon the application (e.g., Nxxxx specifies a parts program line number "N101", Txxxx specifies the number of the cutting tool that is being used)
- MFO is an acronym for Manual Feed Override
- MSO is an acronym for Manual Speed Override
- CNC is an acronym for Computerized Numerical Controller
- MDI is an acronym for Manual Data Interface
- This manual uses the symbol "∇ ∇ ∇" to indicate the end of a chapter.

Program lines (or blocks) are strings of characters terminated by a carriage return or line feed, or both.

Program lines consist of program words, each separated by white space. This white space is a string of any number of spaces, commas or tabs. For example, the following lines are reasonable equivalents.

```
G1,, X7
G1 X7
```

In many cases white space is not needed (for example, G1X7 is OK), but it is recommended because the CNC may not be able to decipher the words properly without white space.

The programming commands presented in this manual are grouped according to their mode of operation. Most commands appear on an individual page with an example of their usage following it.

G-codes, M-codes, and extended commands appear in **bold face** letters within their respective chapters.

All reserved words (i.e., axis names, commands, etc.) may not be used as variables, labels or soft axis names.

Although every effort has been made to ensure consistency, subtle differences may exist between the illustrations in this manual and the component and/or software screens that they represent.



CHAPTER 1: SYMBOLS & AXIS DESIGNATORS

In This Section:	
• Special Symbols	1-1
• Special Characters	1-3
• Parameter Types	1-8
• Variables	1-9
• Tool Word	1-19
• Multiple CNCs	1-22

1.1. Special Symbols

Some symbols used with the UNIDEX 631/U600 are reserved symbols that have special purposes. Their function is to allow the user, when programming, to insert comments about a particular program block. These comments are ignored during program execution. They also permit the user to omit specific program blocks. Besides allowing the user to ignore or omit program syntax, special symbols will permit the user to extend program blocks.

1.1.1. Comment Operator ;

The UNIDEX 631/U600 CNC interprets the percent sign and the semicolon symbols as the start of a comment. All text following these characters (on the same line) will be ignored when executing the program.

These operators are commonly used to document a parts program. This is useful when later modifying the program. Any textual .INI file can use comments as well, specifically, AUTORUN.INI, GLBALIAS.INI, OCC1.INI, MODSCAN.INI, and MCODEx.INI.

1.1.2. Block Delete Character /

The UNIDEX 631/U600 CNC provides a feature referred to as block delete which permits certain program blocks to be omitted during program execution. This feature is activated/deactivated using the <BLOCK DELETE> control located on the CNC run screen (refer to the UNIDEX 631/U600 User's Manual).

When used as the first character of a program block, the "/" symbol is used to designate that this line is subject to be conditionally executed, dependent upon the current state of the block delete function.

When block delete is active, designated program blocks are treated as comments, and therefore omitted from execution. When this feature is inactive, these program blocks are executed normally. By default, all program blocks are executed.

1.1.3. Start Extended Command Block (

When used as the first character of a program block, the "(" character designates the start of an extended (non RS-274) command block. Since several of these commands may span multiple command blocks, the end of extended command block operator is needed to terminate the block. The end of an extended command block operator is the right parenthesis character ")" bracket.

1.1.4. End Extended Command Block

When used as the last character of a program block, the ")" character designates the end of an extended (non RS-274) a command block. As mentioned above, several of these commands may span multiple command blocks, and therefore, the end extended command block operator is needed to terminate the block.

1.1.5. Array Indices

As described below, the UNIDEX 631/U600 supports operations on arrays, or groups of CNC variables. To specify an array element, you must specify an index to the desired element. This is done by using the "[" and "]" characters to delimit the index. Refer to Chapter 4, Extended Commands, for explanation on defining arrays using the DVAR command. However, global variables may be addressed as arrays without any special declaration.

The first element of an array is indexed as 0, for example, if the array MyArray is defined to contain 10 elements, the sixth element of the array would be designated as follows.

MyArray[5]



A check of the index validity is not performed. If the index is outside the range defined for that array (as defined in the DVAR command), the syntax will specify other variables, and this may lead to unexpected results.

No math is allowed within the subscripts. However, variables may be used even with array indices. But the array references can only be two deep (see examples below).

SYNTAX:

MyArray [10]
or
MyArray [MyVariable]
or
MyArray [MyVariable [3]]
or
GLOBAL10 [5]

Any variable can be indexed, even local variables not declared in a DVAR statement, and the index is used as an offset from the actual variable storage location. Indexes intended use are with local arrays (refer to Chapter 4, DVAR extended command), but can be used in any variable. For example, GLOBAL1[5] refers to the variable "GLOBAL6". Other examples are shown below.

(DVAR vara, varb[5], varc)

where

varb[-1] refers to *vara*
varb[5] refers to *varc*
vara[1] refers to *varb[0]*

In the above example, *vara[-1]* and *varc[1]* refer to non-existent variables and their usage can lead to unexpected results.



1.2. Special Characters

The UNIDEX 631/U600 CNC supports two methods of specifying axis names, hard names and soft names. Soft names refer to the name specified in the machine parameter "Axis Name." Hard names refer to a pre-defined set of letters that correspond with specific axes. The set of hard names used by the UNIDEX 631/U600 is defined as follows.

Table 1-1. Hard Axis Names

Hard Name	Axis Number	Hard Name	Axis Number
X, Y, Z	1-3	x,y,z	10-12
U, V, W	4-6	u,v,w	13-15
A, B, C	7-9	a	16

For additional information on axis designations, refer to the hard names and soft names command descriptions (Section 4.8.1. Axis Naming in chapter 4 of this manual).

The soft names entered in the Machine Parameter Menu must be characters A through Z and/or numbers. If a number appears in the name, such as axis1, the number must not be one of the first three characters. No spaces are allowed in the name. Soft names are the default. No reserved letters may be used as a softname, such as G, X, Y, F, I, J, etc.. However, GG, XX, YY, FF, II, JJ, GGG, XXX, etc... are acceptable.

1.2.1. Line Numbers

Nxxxx

When used as the first characters of a program block, the "N" character, followed by any number of digits, may be used to specify the parts program line number. The UNIDEX 631/U600 CNC ignores this line number designation, and therefore, does not force a particular numbering scheme. These N code line labels may not be used as a jump or subroutine entry label.

The "N" code can not be used in a JUMP, DENT, or CALL subroutine statement.



SYNTAX:

N100
N101 (Where the numeric digits specify the parts program line number).

1.2.2. Linear Feedrate

F

The keyword F is used to provide a parameter to various G codes. Its meaning depends on the G code preceding the F word.

- | | |
|-------------------------------------|--------------------------------------|
| 1. G1,G2,G3,G11,G12,G13 | - The F word is a feedrate |
| 2. NONE, (F word appears by itself) | - The F word is a feedrate |
| 3. G4 | - The F word is a time (seconds) |
| 4. G60 through G66 | - See that G code for F word meaning |

All the following text applies only when the F word is a feedrate, any of its other uses when used with G-codes, refer to Chapter 3 under the documentation of the specific G-code being used.



An axis is designated as linear or rotary from within the Machine Parameters Set-up Screen using the "Linear or Rotary" parameter. For details on the usage of these parameters, please refer to the *UNIDEX 631/U600 User's Manual*.

The F word is always used when motion is to be performed on a linear axis. However, if both linear and rotary axes are to be moved within the same program block, the current Dominant Feed (G98/G99) operational mode must be evaluated to determine which feedrate will apply.

The F word retains its value until overwritten by a subsequent program block. Therefore, all G1/G2/G3 moves need not specify an F feedrate explicitly.

During G0 moves, each axis will follow its own rapid feedrate as specified in the Machine Parameters Screen. Some axes may complete their move before other axes, depending on the targets and specified rapid feedrates. In G1/G2 and G3 moves all of the speeds of each axis must be coordinated so that all the axes complete their move at the same time. This is called a contoured move. In a contoured move the programmer specifies only one feedrate; the vector feedrate which the move follows. This is called the F word.

$$\text{F word} = \frac{\text{square root of } (X^2 + Y^2 \dots)}{\text{duration of move}}$$

Where the sum in the numerator is computed for all axes involved in the move and X,Y,etc. are the individual velocities of each axis involved in the move.

However, if the axes to be are rotary axes, (zero to 360 degrees) then the E word is used instead. The same formula for the F word above, applies to the E word. A non-rotary axis is often called a linear axis.

1.2.3. Feedrate Limiting

Contoured moves will never exceed certain maximum feedrates that are specified by the programmer. The controller guarantees that the limit will not be exceeded by scaling down the velocity of the move to the highest speed that does not violate any limit. There are a number of different limits that can exist. They are listed below.

1. Normalcy speed limit See B-axis parameters screen.
(invoked under the CNC Initialization menu item)
2. Rapid traverse axis speed Machine Parameters Screen.
3. Maximum F feedrate CNC Initialization Screen.
4. Programmed E feedrate Set in a CNC line with an E word.

If multiple limits are violated in one move, the move will be scaled down until none of the limits are violated. The four limits listed above are discussed further in the following paragraphs.

The Normalcy Speed limit applies only when normalcy is active (refer to Chapter 2 Normalcy Mode) and is applied only to the normalcy axis. In some cases (the normalcy alignment move) the maximum feedrate and rapid traverse axis speed limits are not applied, only the Normalcy Speed limit is applied. (refer to Chapter 2 Normalcy Mode).

The rapid traverse axis speed limit (also the speed at which G0 moves are performed), is applied on a per axis basis. So, if any axis in a multiple axis move exceeds its own rapid feedrate, the speed of the entire move will be scaled down until that axis is traveling at its rapid feedrate.

The maximum F word feedrate limit is applied to all the linear (non-rotary) axes in the move. The linear vector speed, (i.e. speed along the vector of the linear axes) will never exceed this value. Rotary axes in the move do not enter into this limiting. In the normal case, the programmer cannot even compile a program specifying a F word larger than the maximum, for example if the maximum is F100 and a CNC program line contains "F110", then the program will trigger a compile error. However, there are a number of conditions where the compiler cannot detect that a move is directing a linear feedrate above the maximum F rate and the limiting is applied.

- a. When MFO > %100 (see Section 1.2.2.1)
- b. When a non-dominant axis is moved in a complex move (see Chapter 2 Federate and Spindle Speed Codes)
- c. When variable feedrates are used (see syntax description)

The programmed E word feedrate limit is applied to the rotary axes in the move. The rotary vector speed, (i.e. speed along the vector of the rotary axes) will never exceed this value. Unlike the F word, there is no explicit maximum, the maximum is taken to be the current E word setting. As in the F word, this limit can only be exceeded under the three conditions listed above.

1.2.3.1. Notes on the Feedrates Display Screen

This screen can be found in the manual or program screens, under the Status menu item. It shows the programmed and actual feedrates for the MFO scroll bar and can be varied at anytime, between 0% and 150%. However, it will not exceed 100% for G0 moves. If the user sets it to 0% the move will pause, as if in feedhold. The screen also shows the currently programmed feedrates, (current E and F words) and the actual feedrates. The actual feedrates represent the actual E and F words being achieved in the current move. In a G0 mode, the actual feedrates are always zero, even though the axes are moving. During a contoured move, the move (when not in feedhold or fault) the actual feedrates will differ from the product of the programmed time and the MFO percentage only under the following circumstances.

1. There is a limiting feedrate preventing the programmed feedrate from being achieved and the feedrate is dominant.

Under either of these conditions the actual feedrate number will flash.

SYNTAX: *F\$ Variable.* (If the feedrate is specified within a variable).
 or
 F100. (the value is specified by a literal value).

1.2.4. Rotary Feedrate

E

The keyword E is used to specify a rotary vector feedrate. Note that this has no relation to the Spindle rotary axis speed, which is set with the S word. The units of the E word are always RPM, and unlike the F word, the units are not affected by the G70/G71 or G93/G94/G95 settings. The E word is only used in the G1,G2,G3, ,G12,G13 G-codes moves.

Note that the E word retains its value until overwritten by a subsequent program block. Therefore, all G1/G2/G3 moves need not explicitly specify an E feedrate.

An axis is designated as rotary from within the Machine Parameters Set-up Screen using the "Linear or Rotary" parameter. For additional information on the usage of this parameter, please refer to the *UNIDEX 631/U600 User's Manual*.



This E word. is always used when motion is to be performed on rotary axes only. However, if both linear and rotary axes are to be moved within the same program block, the current Dominant Feed (G98/G99) operational mode must be evaluated to determine which feedrate will apply.

There must be an E feedrate specified for the Spindle Reorient (M19).

The E word is also used as a limit on rotary axis motion. Refer to the notes on limiting under the F word documentation for more details.

SYNTAX: *E\$Variable.* (if the feedrate is specified within a variable).
 or
 E100. (the value is specified by a literal value).

1.2.5. Spindle Feedrate

S

The keyword S specifies the velocity of the spindle axis (in revolutions per minute) when the spindle is under direct program control (G97). When constant surface speed is programmed (G96), this parameter has no effect. However, G96 uses the SF keyword to specify surface feedrate.

It should be noted that this keyword retains its value until overwritten by a subsequent program block. Therefore, all program blocks that turn on the spindle need not specify a feedrate explicitly.

A rotary axis is designated as the spindle axis from within the CNC Initialization Set-up Screen. For more information regarding spindle axis requirements, please refer to the *UNIDEX 631/U600 User's Manual*.



SYNTAX: *S\$Variable.* (if the feedrate is specified within a variable).
 or
 S100. (the value is specified by a literal value).

1.2.6. Constant Surface Speed Feedrate

SF

The keyword SF specifies the velocity of the surface spindle axis (user units per min.) when the surface spindle is under direct program control (G96). When velocity of the spindle axis is programmed (G97), this parameter has no effect. However, G97 uses the S keyword to specify surface feedrate.

SYNTAX: *SF\$Variable.*
 or
 SF100.

1.2.7. Circle CenterPoints

I/J/K

The keywords I, J, and K are used when specifying the centerpoint of a circle during circular interpolation (see G2, G3, G12, or G13). The CNC Parameters Plane Selection Menu is used to assign three axes on which circular interpolation may be performed. These three axes are referred to as the Plane X Axis, the Plane Y Axis, and the Plane Z Axis.

When initiating circular motion, you must describe the radius of the arc by specifying the circle centerpoint. As mentioned, the I, J, K keywords are used for this purpose. I is the relative position of the circle centerpoint with respect to the axis designated as the Plane X Axis in this screen. Similarly, the J and K keywords are associated with the Plane Y Axis and the Plane Z Axis, respectively.

For more information on the usage of the I/J/K keywords, refer to the "G2" and "G3" command descriptions (Motion G-codes in chapter 2 of this manual). Also, refer to G-codes G2, G17 through G19, and G110.

1.3. Parameter Types

G-codes, M-codes, and extended commands often require or allow parameters to be entered on the same block. This section covers the details regarding the syntax for certain parameters.

1.3.1. Floating Point Constants

The U631/U600 supports double precision (64 bit), where floating point values are tracked to 15 digit accuracy. In other words, the value "123456789.012345" is recognized as a different value compared with the value "123456789.012344". But is not recognized as different than the value "123456789.0123451". The following examples are all valid and represent the value 1.75.

- 1.750
- .00175e³
- 175e⁻²

1.3.2. Character Strings

The Character strings supported by the U631/U600 must be less than 40 characters in length. Some extended commands accept character strings.

Character strings are always surrounded by double quotes (“”).



Example: “*This is a string*”

1.3.3. Filenames

Some extended commands accept filenames as parameters. The total length of the string must be less than 40 characters. The path and type can be provided in the filename.

Example:

C:\U31\Programs\data.dat

Do not surround filenames with double quotes.



1.3.4. Hexadecimal Integers

A hexadecimal number specification up to 8 digits long may be specified by preceding the number with the “#” sign.

Example: `var1 = #1AF0`

1.4. Variables

Variables or variable names can be used anywhere floating point constants are used. The types of variables used are local, static, global, and system. Variable names must be less than 20 characters in length and the value of a variable is resolved at “run time” when a particular CNC command using that variable is executed.

All variables are double precision and have a range of $1.7 \text{ E } +\text{-} 308$ (1.7×10^{-308} through $1.7 \times 10^{+308}$) with 15 digit accuracy.

Each element of an array counts as one variable.



1.4.1. Local Variables

Local variables must be defined in a program with the extended command “DVAR” and can only be used within the program in which they are defined. Also, the user can reserve arrays of local variables with “DVAR” (see Chapter 4, Extended Commands).

Example: (DVAR A, B)
 A = 5

1.4.2. Global Variables

Global variables are variables which may be accessed by any of the four CNC engines present in the UNIDEX 631/U600. Variables of this type are very useful for communication between parts programs executing independently on the 4 independant CNC engines.

The number of global variables available for use is defined within the Variable Allocation Group box of the CNC Initialization Screen (refer to the *UNIDEX 631/U600 User’s Manual*). It should be noted that this number is applicable to all CNC engines which are active at any given time.

Normally, all global variables are initialized to zero on program startup. However, if the save globals option is selected (on the Options screen) then global variables are written to disk upon program exit, and restored on the next program restart. Furthermore, the Global Alias file can be used to effect global variable defaults (see section 1.4.3).

SYNTAX: GLOBALxx (where xx refers to the variable number).

EXAMPLES:

(DVAR, VAR1)	;Define the local variable VAR1
VAR1 = GLOBAL00	;Read the value of global variable #00
GLOBAL99 = 5	;assign the value of global variable #99 to 5



There may be a maximum of 1,000 global variables.



Even if the save globals option is selected, and if the GLBALIAS.INI file indicates a default value (see section 1.4.3) then the default value will be assigned to the variable.

1.4.3. Using the Global Alias File

Global alias files allow the user to provide an alternate name for pre-defined global variables to a variable name chosen by the user that has meaning in the context of its usage. The default keyword allows the user to set non-zero default values (upon CNC initialization) for global variables.

An optional EDIT keyword allows the user to edit the value of the global variable at run time. If this keyword is specified, an optional comment text string displayed in the global variable edit window may also be specified. The EDIT keyword allows a valid range for the variable to be set by the MIN and MAX keywords. This prevents the user from entering a bad value.

Figure 1-1 shows an example of how to accomplish alternate naming of pre-defined variables. Variables given an alias can still be referred to by the standard means, i.e., MASTER_CORRECTION is identical to GLOBAL0.

;	
;	Global Alias .INI Sample File (GLBALIAS.INI)
;	
0 MASTER_CORRECTION	; Basic alias definition for GLOBAL0 variable
1 MAX_SPEED DEFAULT 100	; Basic alias definition with a DEFAULT value
2 START_SPEED EDIT " deg/sec"	; Alias that may be EDITed by user at run time
	; (with a comment)
3 MAX_SIZE EDIT	; Alias that may be EDITed by user at
	; run time (withOUT a comment)
4 ACCEL_TIME MIN 5 MAX 200 EDIT "milliseconds "	; Alias that may be EDITed by user at
	; run time (with a comment) & range specified
5 DECEL_TIME DEFAULT 7 MIN 5 MAX 200 EDIT "milliseconds "	; Alias that may be EDITed by user at run time
	; (with a comment) & range specified & DEFAULT
	; value

Figure 1-1. Renaming Variables

SYNTAX: # Alias [DEFAULT val][EDIT ["Global variable comment"] [min val max val]]

where # is the number of the global variable

The global alias file is GLBALIAS.INI and is stored in the INI directory. For additional details on using the global alias file, read the comments contained within the GLBALIAS.SAM file located in the \U31\INI directory.

Global variables whose values are not set as a default and the save global options is not active, will have their value set to zero upon CNC initialization.



1.4.4. Static Variables

STATICxx

Static variables are variables which may be accessed by all of the parts programs executed on a given CNC. Variables of this type are very useful for communication between parts programs executing within a given CNC.

The number of static variables available for use is defined within the Variable Allocation Group box of the CNC Initialization Screen (refer to the *UNIDEX 631/U600 User's Manual*). It should be noted that this number is applicable to all CNC engines which are active at any given time.

SYNTAX: *STATICxx* (where xx refers to the variable number).

EXAMPLE:

(DVAR, VAR1)	;Define the variable VAR1
VAR1 = STATIC00	;Read the value of the static variable #00
STATIC69 = 5	;assign the value 5 to the static variable #69



Upon CNC initialization, all static CNC variables are set to zero. The value of all static CNC variables are lost upon loading a new program and executing it from the RUN menu of the CNC application.

Variables of this type are used for communication with other parts programs executed with the CALL and EXECUTE command discussed in Section 4.4.

There may be a maximum of 1,000 static variables.

1.4.5. System Variables

The UNIDEX 631/U600 CNC provides the ability to access information about the environment in which the program is executing. This information is provided via a set of read-only variables which are pre-defined for use in any parts program.

Due to the similarity in syntax, specific examples will not be shown for each variable. In all cases, the following usage is applicable.

VARI = SystemVariable

1.4.5.1. CNC Number CNCNUM

This system variable contains a value corresponding to the number of the CNC engine that is executing this parts program. The range of this value is from 0 through 3 that represents CNC 1 through 4 and remains constant throughout program execution.

1.4.5.2. CNC Time CNCTIME

This system variable always contains the number of milliseconds for which the CNC has been active. Upon CNC initialization, this variable is set to zero and is incremented every millisecond. It may be used as a time base within individual parts programs.

1.4.5.3. CNC Position Command POSITNxx

The variables POSITN01 through POSITN16 are used to access the current position command for the axes. For example, XXX = POSITN06, this assigns the current position command for axis 6 to the variable XXX.

“06” is the two digit axis specifier. All axis must be specified with two digits.



1.4.5.4. CNC Preset Command PRESETnn

The CNC variables PRESET01 through PRESET16 are used to access the CNC's preset position registers that may be set by the G92 (software home) command. For example, var = PRESET01 will return the preset position of the X axis into the variable “var.”

The 2 digit axis specifier “01,” all axes must be specified with 2 digits.



1.4.5.5. CNC Status Words

CNCSTATx

The variables CNCSTAT1, CNCSTAT2, CNCSTAT3 and CNCSTAT4 are used to access information about the current motion status of the CNC. Each of these status words are 32 bit words where each bit reflects a unique piece of information. A one in a corresponding bit shows the condition is active. Inactive conditions are represented by a zero.

These variables can be used to communicate CNC process status to another CNC engine, via global variables or to other equipment in the system, such as a remote host or a programmable logic controller. Table 1-2 defines the CNCSTAT1, CNCSTAT2, and CNCSTAT3 variables.

The lowest 16 bits of each of these status words are mapped into virtual I/O in the range of 448 to 512. See example below.

Example:

CNCSTAT1 Bit 0 is mapped to 448..... Bit 15 is mapped to 463
CNCSTAT2 Bit 0 is mapped to 464..... Bit 15 is mapped to 479
CNCSTAT3 Bit 0 is mapped to 480..... Bit 15 is mapped to 495
CNCSTAT4 Bit 0 is mapped to 496..... Bit 15 is mapped to 511

Table 1-2. CNCSTAT1, CNCSTAT2, and CNCSTAT3 Variables

/* CNC status word #1		
#define CNC_ACTIVE	0x00000001	/* CNC running */
#define CNC_SPINDLE_ACTIVE	0x00000002	/* Spindle is active*/
#define CNC_INITIALIZED	0x00000004	/* CNC initialized */
#define CNC_STOPPING	0x00000008	/* CNC is stopping*/
#define CNC_MOTION	0x00000010	/* CNC is doing motion*/
#define CNC_AXIS_FAULT	0x00000020	/* Axis fault */
#define CNC_SINGLE_STEP_MODE	0x00000040	/*single step mode */
#define CNC_SINGLE_STEP_HOLDING	0x00000080	/*single step holding */
#define CNC_HOLDING	0x00000100	/* hold mode */
#define CNC_HOLDING_NOW	0x00000200	/*holding now */
#define CNC_FEEDRATE_LOCK	0x00000400	/*feedrate lock*/
#define CNC_SPINDLE_FEEDRATE_LOCK	0x00000800	/* spindle feedrate lock */
#define CNC_MANUAL_MODE	0x00001000	/* Spindle direction is negative */
#define CNC_HOLD_INPUT	0x00002000	/* hold input active */
#define CNC_ESTOP_INPUT	0x00004000	/* manual mode selected */
#define CNC_FAULT	0x00008000	/* CNC fault */
#define CNC_SPINDLE_NEGATIVE	0x00010000	/* Linear accel/decel mode */
#define CNC_INTERRUPT_ENABLE	0x00020000	/* CNC interrupts enabled*/
#define CNC_INTERRUPT_FEEDHOLD	0x00040000	/* interrupt feedhold active*/
#define CNC_BLOCK_DELETE_ENABLE	0x00080000	/* block delete enabled */
#define CNC_OPTIONAL_STOP_ENABLE	0x00100000	/* optional stop enabled */
#define CNC_DEBUG_MODE	0x00200000	/* Debug mode*/
#define CNC_NEW_LINE	0x00400000	/* when a new line is executed*/
#define CNC_HOME_MOTION	0x00800000	/* home motion */
#define CNC_IMMEDIATE_MCODE	0x01000000	/* immediate mcode execution active */
#define CNC_G0_MOTION	0x02000000	/* G0 motion */
#define CNC_PROBE_ENABLE	0x04000000	/* Probe is enabled */
#define CNC_CANCEL_MOVE	0x08000000	/*
#define CNC_RETURN_MOVE	0x10000000	/*
#define CNC_SPLINE_MOVE	0x20000000	/* G30 - splinning active */
#define CNC_THREADCUTTING	0x40000000	
#define CNC_IMMEDIATE_GCODE	0x80000000	

Table 1-2. CNCSTAT1, CNCSTAT2, and CNCSTAT3 Variables Cont'd

/* CNC status word #2		
#define CNC2_ACCEL_RATE	0x00000001	/* rate based acceleration mode*/
#define CNC2_DECEL_RATE	0x00000002	/* rate based deceleration mode*/
#define CNC2_MASTER_HOLD	0x00000004	/* ored hold input */
#define CNC2_VARIABLES_ALLOCATED	0x00000008	/* CNC variables allocated */
#define CNC2_ALLOCATED	0x00000010	/* CNC memory allocated */
#define CNC2_OPTIONAL_STOP_HOLDING	0x00000020	/* optional stop holding */
#define CNC2_NORMALCY_MOVE	0x00000040	/* doing normalcy move */
#define CNC2_CLEANUP	0x00000080	/* cleanup in progress */
#define CNC2_REQUEST_PROGRAM_LOAD	0x00000100	/* request program load */
#define CNC2_G9	0x00000200	/* G9 mode of last motion block*/
#define CNC2_REQUEST_PROGRAM_EXECUTE	0x00000400	/* request program execution */
#define CNC2_DISPLAY_CDW_TEXT	0x00000800	/* new text line to be displayed */
#define CNC2_CDW_OPENED	0x00001000	/* CDW Opened */
#define CNC2_CDW_RECORD_ON	0x00002000	/* CDW record on */
#define CNC2_REQUEST_CDW_RECORD_ON	0x00004000	/* request CDW record on */
#define CNC2_M00_HOLDING	0x00008000	/* */
#define CNC2_SPLINE_CALCULATED	0x00080000	/* spline coefficients calculated */
#define CNC2_INTERRUPT_PENDING	0x00100000	/* interrupt pending */
#define CNC2_QUEUE_ALLOCATED	0x00200000	/* continuous block mode*/
#define CNC2_REQUEST_TWORD	0x00400000	/* requesteing TWORD information*/
#define CNC2_ON_ERROR_ACTIVE	0x00800000	/* on error routine active*/
#define CNC2_ON_ERROR_POSTED	0x01000000	/* on error posted to cncslice */
#define CNC2_QUEUE_EMPTY_HOLDING	0x02000000	/* holding because queue is empty*/
#define CNC2_REQUEST_FILE_OPEN	0x04000000	/* requesting file open */
#define CNC2_REQUEST_FILE_CLOSE	0x08000000	/* requesting file close */
#define CNC2_REQUEST_FILE_RESET	0x10000000	/* requesting file reset */
#define CNC2_REQUEST_FILE_READ	0x20000000	/* requesting file read */
#define CNC2_REQUEST_FILE_WRITE	0x40000000	/* requesting file write */
#define CNC2_REQUEST_CALLBACK	0x80000000	/* requesting a call back statment recompute */
/* CNC status word #3		
#define CNC3_ON_ERROR_RUNNING	0x00000001	/* ON ERROR running */
#define CNC3_MONITOR_SPEED	0x00000002	/* monitor speed on */
#define CNC3_LAST_MONITOR_SPD_VALID	0x00000004	
#define CNC3_REQUEST_CDW_CLOSE	0x00000008	/* request CDW close */
#define CNC3_REQUEST_DATACOLLECT_OPEN	0x0040	/*request data collection window to open */
#define CNC3_REQUEST_DATACOLLECT_CLOSE	0x0080	/* request data collection window to close */
#define CNC3_DATACOLLECT_OPEN	0x0100	/*data collection has been opened */
#define CNC3_DATACOLLECT_START	0x0200	/*data collection is active */
#define CNC3_CDW_IRQ_PENDING	0x400	

1.4.5.6. CNC Operational Mode

The system variables CNCMODE1, CNCMODE2, CNCMODE3 and CNCMODE4 are used to show information regarding the current operational mode of the CNC. This modal information, such as G70/G71 and G90/G91, is stored as a 32 bit word in which each bit reflects a different piece of information. A one in a corresponding bit shows the condition is active. Inactive conditions are represented by a zero.

These variables can be used to communicate the status of a CNC process to another CNC engine, via global variables or to other equipment in the system, such as a remote host or a programmable logic controller. Table 1-3 defines the CNCMODE1 and CNCMODE2 variables.

Table 1-3. CNCMODE1 and CNCMODE2 Variables

/* CNC Mode 1 bits defined		
#define CNC_MODE_G90	0x00000001	/* absolute if set*/
#define CNC_MODE_G91	0x00000002	/* incremental if set*/
#define CNC_MODE_G70	0x00000004	/* English if set*/
#define CNC_MODE_G71	0x00000008	/* Metric if set*/
#define CNC_MODE_G41	0x00000010	/* Cutter Comp Left */
#define CNC_MODE_G42	0x00000020	/* Cutter comp Right*/
#define CNC_MODE_G54	0x00000040	/* fixture offsets #1*/
#define CNC_MODE_G55	0x00000080	/* fixture offsets #2 */
#define CNC_MODE_G0	0x00000100	/* G0 active */
#define CNC_MODE_G1	0x00000200	/* G1 active */
#define CNC_MODE_G2	0x00000400	/* G2 active */
#define CNC_MODE_G3	0x00000800	/* G3 active */
#define CNC_MODE_G4	0x00001000	/* G4 active */
#define CNC_MODE_G8	0x00002000	/* G8 active */
#define CNC_MODE_G9	0x00004000	/* G9 active */
#define CNC_MODE_G93	0x00008000	/* inverse feedrate */
#define CNC_MODE_G94	0x00010000	/* normal feedrate */
#define CNC_MODE_G95	0x00020000	/* G95 active */
#define CNC_MODE_G96	0x00040000	/* G96 active */
#define CNC_MODE_G97	0x00080000	/* G97 active */
#define CNC_MODE_G40	0x00100000	/* G40 active */
#define CNC_MODE_G30	0x00200000	/* G30 active */
#define CNC_MODE_G101	0x00400000	/* Manual Mode*/
#define CNC_MODE_G102	0x00800000	/* Auto Mode */
#define CNC_MODE_G110	0x01000000	/* normal g2 and g3 if set */
#define CNC_MODE_G111	0x02000000	/* Reverse g2 and g3 if set */
#define CNC_MODE_G67	0x04000000	/* accel/decel time mode */
#define CNC_MODE_G68	0x08000000	/* accel/decel rate mode */
#define CNC_MODE_G21	0x10000000	/* Normalcy On Left */
#define CNC_MODE_G22	0x20000000	/* Normalcy On Right */
#define CNC_MODE_G56	0x40000000	/* Monitor parm mode */

Table 1-3 CNCMODE1 and CNCMODE2 Variables Cont'd

/* CNC Mode2 bits defined		
#define CNC_MODE2_G83	0x00000001	/* Mirror active */
#define CNC_MODE2_G84	0x00000002	/* rotate active*/
#define CNC_MODE2_G98	0x00000004	/* Rotary Dominant */
#define CNC_MODE2_G99	0x00000008	/* Linear Dominant */
#define CNC_MODE2_SPINDLE_SHUTDOWN	0x00000010	/* kill spindle when M02*/
#define CNC_MODE2_ANALOG_MFO	0x00000020	/* Analog MFO active */



The system variables CNCMODE2, CNCMODE3 and CNCMODE4 are reserved for future expansion.

1.4.6. CNC Analog Feedback Commands

With the U631 the analog inputs from the optional MATRIX eight channel analog card may be read from within a CNC program using the following commands:

```
ANALOG0
ANALOG1
ANALOG2
ANALOG3
ANALOG4
ANALOG5
ANALOG6
ANALOG7
```

With the U600 the four on-board analog channels may be read with the *ANALOG0* - *ANALOG3* keywords. The four analog inputs on the optional encoder expansion board 1 would be read with *ANALOG4* through *ANALOG7* keywords. Encoder expansion board 2 would be read with *ANALOG8* through *ANALOG11* keywords and encoder expansion board 3 would be read with *ANALOG12* through *ANALOG 15* keywords.



The U600 analog channel 0 is the MFO input if enabled through the software, channels 2 and 3 are the X and Y pots of the optional joystick.

These commands may be used in the same manner that the CNCMODE, CNCNUM, and CNCSTATUS commands are used (refer to Systems Variables, Section 1.4.5.). An example of their use is shown below.

```
GLOBAL2=ANALOG0
or
GLOBAL3=ANALOG0/3276.8*.005
```


1.5. Tool Word

Txxxx

The keyword T specifies the number of the cutting tool which is being used. This number is used to search the tool file associated with this CNC to determine all information required about a specific tool. Please refer to the section entitled *UNIDEX 631/U600* CNC Tool Handling for more information on the operation of this feature.

1.5.1. UNIDEX 631/U600 CNC Tool Handling

The tool handing feature of the *UNIDEX 631/U600* CNC is based upon a file referred to as the Tool File (TF). This file contains information on all the currently available tools in a machining station. Currently, a maximum of 18 tools may be present in any machining station. However, future releases will support the use of up to 30 tools per station. This tool information is used when Intersectional Cutter Radius Compensation is in effect.

As mentioned, the tool file contains one record for each tool currently available. Each tool record contains seventeen different fields. Four of these are reserved for future expansion. The information found in these fields will be read by the axis processor each time the cutting tool is changed (using the T keyword). Specifying tool T0000 deactivates all tools from the system.

All of this information may be viewed/modified using the Tool file editor option of the Set-up menu. Please refer to the *UNIDEX 631/U600 User's Manual* for more information on the operation of this dialog box.

As mentioned, each tool record contains 17 entries, 4 of which are reserved. The following discussion describes each of the 13 fields which are currently in use.

1. Radio Frequency Tool Identification Number

This number is a unique number used to distinguish between the various tools available at this station. It is specified immediately following the T keyword to activate the tool parameters. It should be noted that T0000 is used to deactivate all tool offsets.

2. Tool Type

This field contains an indication of the type of the tool. Valid values include 1 through 4, and are defined as follows:

1	Turning Tool
2	Touch Probe
3	Cleaning Tool
4	Drilling Tool.

3. Tool Location

This field is used to denote the current location of the tool within a station. Twenty possible locations exist:

1-18	Location on tool carousel
19	In the tool changer gripper
20	In the tool post.

4. Tool Orientation

The tool orientation defines the orientation of the tool with respect to its placement in the tool post. Valid values for this field include the following:

1	Right hand
2	Left hand
3	Upside-down right hand
4	Upside-down left hand.

5. Nominal Radius

This field contains the tool radius as specified by the tool manufacturer. This value must be manually updated by the operator.

6. Actual Radius

This field contains the tool radius as measured by the machine operator. This value must be manually updated by the operator.

7. Number Passes

This field is used to specify the number of passes a tool makes in machining a workpiece. This field must be updated manually by the operator.

8. X Axis Offset

9. Z Axis Offset

The X and Z axis offset values are a means by which the parts program can shift the coordinate system to compensate for the differences between tools. The axes positions in a parts program represent the desired tool path of the axes with no offsets. As the characteristics of the tool change, the actual tool path will change, and it will not be at the desired location. The X and Z axis offset feature allows the same parts program to be used for different tools.

The X and Z axis offset values are interpreted as a signed incremental distances that are added to the preset axes positions. These offset values define the distance to the center line of the tool's radius. The Z axis offset is measured from the centerline of the boring bar to the center line of the tool's radius in the Z direction. The X axis offset is measured from the face of the tool post to the center line of the tool's radius in the X direction.

These fields must be updated manually by the operator.

10. Boring Bar Number

This field contains a unique number used to identify a specific boring bar.

11. Tool Holder Number

This field contains a unique number used to identify a specific tool holder.

12. System Entry Date

This field contains the date at which the tool was made available to the machining position. The format for this field is as follows:

dd/mm/yyyy

where dd is the day
mm is the month and
yy is the year.

13. System Entry Time

This field contains the time at which the tool was made available to the machining position. This time is assumed to be on the System Entry Date. The format for this field is as follows:

HH:MM:SS

where HH is the hours
MM is the minutes
SS is the seconds.

1.5.1.1. Inspection Station Probe Set Data

The inspection station probe data contains the probe tip radius and the U and W axis offset values of the probe on the inspection station. These values may be set within the Station Probe Set Data option of the CNC Initialization Set-up menu. Please refer to the *UNIDEX 631/U600 User's Manual* for more information on the operation of this dialog box.

1.6. Multiple CNCs

The U631/U600 has four independent CNCs where the user can run programs, execute manual mode commands, or jog axes.

The axis processor “timeshares” or splits its computation time equally between the four CNCs to allow the user to run multiple CNCs simultaneously. For example, the user can have an I/O scanning program running on CNC number 1, a program executing on CNC number 3, while executing manual mode commands on CNC number 4.

The only restriction is each CNC must have exclusive control of the axes it owns. In other words, if CNC number 1 is controlling axes X and Y, no other CNC can access data from, or run motion on axes X and Y. However, there are exceptions and they are discussed in the following sections.

1.6.1. CNC Intercommunication - Global Variables

Each CNC runs its program independently from the other CNCs. Each CNC has its own local and static variables that no other CNC can change or reference. However, CNCs do share global variables. Any CNC can set or retrieve the value of any global variable at any time. Therefore, global variables can be used to communicate information between the CNCs. For example, CNC number 2 could be executing a “forever loop” waiting for the value of GLOBAL1 to be 1, where it would shut down all drives. Then, if at any time another CNC sets the value of GLOBAL1 to 1, CNC number 2 will shut down all the drives.

1.6.2. CNC Intercommunication - CNC Parameters Setup

Another way to communicate between the CNCs is through axis parameters. A CNC can perform a (SETPARAM) or (GETPARAM) extended command on any axis as long as one of the CNCs owns that axis. Each CNC is initialized and assigned axes through the “CNC General Parameters” option. Refer to the *UNIDEX 631/U600 User’s Manual* for initializing and assigning CNC axes.

1.6.3. Autorun Mode (Autorun.ini)

Normal CNC execution of a program begins when the user selects and runs a program from the Run screen. However, using the AUTORUN.INI file, programs can be run and sequenced automatically. For example, the user may specify that CNC 2 always run a “forever loop” program that monitors particular safety conditions.



Refer to the comments in the AUTORUN.SAM file for additional examples and explanations.

```
; AUTORUN.INI
;
; RECORD SYNTAX: cnc number, filename, run mode
;
; Each record starts a run screen with the given program on the given CNC. The records
; are read sequentially; after a CNC's run screen closes, the next record calling out
; that CNC will be executed. The records for each CNC are handled independantly;
; launching of a particular CNC's records does not wait for the completion of
; any other CNCs records.
; There are two special flag values that can be specified instead of a cnc number.
; Special flags are executed ONLY AFTER ALL CNCs ARE DONE WITH ALL
PRECEDING RECORDS.
; A flag of 0 ends mainmenu, a flag of 100 or greater rewinds the file and
; repeats it. It will be repeated (x-100) times, where x is the flag value.
; The program and run mode are ignored when the cnc is a special flag.
; FOR SINGLE CNC BUILDS, THE CNC NUMBER IS IGNORED (ALWAYS runs
on cnc #1)
;
; The run mode is a sum of values (bit mapped) as follows:
; 1 - automatically run the program if it compiles.
; 2 - start up in step mode
; 4 - start up in optional stop mode on
; 8 - start up in block delete on mode
; 16 - run invisible ( no run screen display)
; 32 - exit run screen when (if) program halts.
; 64 - exit run screen when (if) program compiles OK
; 128 - exit run screen when (if) program compiles with errors
; 256 - open recording file for compile messages
;
; example: 2,test1,6 ; start test1.pgm on cnc#2, with step and optional stop on
;
```

Figure 1-2. The Autorun.ini File

▽ ▽ ▽

CHAPTER 2: G-CODES

In This Section:	
• Description	2-1
• Introduction to G-code Motion	2-5
• Motion G-codes.....	2-7
• Plane Selection Codes	2-21
• Normalcy Mode (G20, G21, G22)	2-22
• Safe Zones (G36, G37)	2-26
• Intersectional Cutter Radius Compensation (ICRC) Overview	2-28
• Fixture Offset (G53, G54, G55)	2-39
• Parameter Monitoring (G56, G57)	2-41
• Acceleration/Deceleration Overview (G60, G61)	2-42
• Programming Codes	2-50
• Feedrate and Spindle Speed Codes	2-59
• Dominant Feedrate Overview (G98, G99)	2-65
• Circular Direction Codes.....	2-69

2.1. Description

A G-code executes motion or is a preparatory function used to specify a particular mode of operation during later motion. In some cases, the new operational mode is applicable to only the particular parts program block in which the G-code is encountered. In other cases, the new operational mode is retained as the default for subsequent parts program blocks.

G-codes of the second type, which permanently change the operational mode of the controller, are referred to as "Modal." The symbol shown in the margin opposite this paragraph will mark all "Modal" commands discussed in this chapter. From this it follows that those which are applicable to a single parts program block are referred to as "Non-Modal."



As can be seen from Table 2-1, on the following page, the G-codes supported by the UNIDEX 631/U600 CNC have been divided into several different functional groups.

Table 2-1. G-code Summary

G-code	Page	Description	Group	Modal
G0	2-7	Rapid Traverse Point-to-point	Motion	Y
G1	2-8	Linear Interpolation	Motion	Y
G2	2-8	CW Circular Interpolation	Motion	Y
G3	2-9	CCW Circular Interpolation	Motion	Y
G4	2-10	Dwell		N/A
G8	2-12	Instantaneous Acceleration		N
G9	2-14	Force Deceleration to Zero Velocity		N
G12	2-16	CW Circular Interpolation	Motion	Y
G13	2-16	CCW Circular Interpolation	Motion	Y
G17	2-21	X/Y Plane Selection Set #1	Plane Select	Y
G18	2-21	Z/X Plane Selection Set #1	Plane Select	Y
G19	2-21	Y/Z Plane Selection Set #1	Plane Select	Y
G20	2-24	Disable Normalcy Mode	Normalcy	Y
G21	2-24	Normalcy On Left	Normalcy	Y
G22	2-25	Normalcy On Right	Normalcy	
G30	2-17	Spline Move		Y
G33	2-18	Thread Cutting		Y
G36	2-26	Enable Safe Zones	SafeZones	Y
G37	2-27	Disable Safe Zones	SafeZones	Y
G40	2-30	Disable Cutter Compensation	ICRC	Y
G41	2-31	Enable Cutter Compensation Right	ICRC	Y
G42	2-33	Enable Cutter Compensation Left	ICRC	Y
G43	2-35	Set Cutter Compensation Radius		Y

Table 2-1. G-code Summary Con't

G-code	Page	Description	Group	Modal
G44	2-35	Set Cutter Compensation Axes		Y
G45	2-36	Disable Polar/Cylindrical Coordinates		
G46	2-36	Activate Polar Coordinates		
G47	2-37	Activate Cylindrical Coordinates		
G53	2-39	Cancel Fixture Offset	FixtureOffset	Y
G54	2-39	Set Fixture Offset #1	FixtureOffset	Y
G55	2-40	Set Fixture Offset #2	FixtureOffset	Y
G56	2-41	Enable Parameter Monitoring		Y
G57	2-41	Disable Parameter Monitoring		Y
G60	2-43	Set Acceleration Time		Y
G61	2-44	Set Deceleration Time		Y
G62	2-45	Set Profile Time		
G63	2-46	Sinusoidal Acceleration Mode	Accel Mode	Y
G64	2-46	Linear Acceleration Mode	Accel Mode	Y
G65	2-47	Set Acceleration Rate		Y
G66	2-48	Set Deceleration Rate		Y
G67	2-49	Acceleration/Deceleration Time Based	Ramp Type	Y
G68	2-49	Acceleration/Deceleration Rate Based	Ramp Type	Y
G70	2-50	English Programming Mode (in.)	Units	Y
G71	2-51	Metric Programming Mode (mm.)	Units	Y
G82	2-51	Clear G92 (software home) Command		
G83	2-52	Mirror		
G84	2-54	Rotate		
G90	2-56	Absolute Programming Mode	Distances	Y
G91	2-57	Incremental Programming Mode	Distances	Y
G92	2-58	Software Home (preset positions)		N/A
G93	2-59	Inverse Feedrate Mode	FeedrateMode	Y
G94	2-60	Normal Feedrate Mode	FeedrateMode	Y
G95	2-61	Inverse Spindle Feedrate Mode	FeedrateMode	Y
G96	2-63	Constant Surface Speed	SpindleSpeed	Y

Table 2-1. G-code Summary Cont'd

G-code	Page	Description	Group	Modal
G97	2-64	Direct Spindle	SpindleSpeed	Y
G98	2-67	Rotary Feedrate Dominant		
G99	2-68	Linear Feedrate Dominant		
G110	2-69	Normal Circular Interpolation	CircleDir	Y
G111	2-70	Reverse Circular Interpolation	CircleDir	Y
G130	2-71	4 Kilo-Hertz Servo Update Rate		
G131	2-71	1 Kilo-Hertz Servo Update Rate		

During initialization, the UNIDEX 631/U600 CNC activates one mode from each operational group. These modes are known as the default operational. However, in many cases, the default operational mode may be changed via the G-codes Menu of the CNC Initialization screen.

In general, a particular parts program block may contain multiple G-codes. However, in order to avoid contradictions, only one G-code from a particular group may appear on a given parts program block.

If multiple G-codes from a group appear on one line, the last one in the line will be the one in effect.

The programmer should also be aware that some extended commands also generate motion (INDEX, STARTM).

2.2. Introduction to G-code Motion

The remainder of this introduction serves three purposes:

1. A guide to the G-code documentation under which various topics of CNC motion are completely explained.
2. An introduction to CNC directed motion.
3. A repository for miscellaneous information not contained under any G-code documentation section.

Further elaboration on the following topics of interest can be found discussed under the documentation section shown to the right of the topic.

Position	(see below)
Velocity	(see below and F code)
Acceleration	(see G60 or G61)
Transition between motion blocks	(see G8 or G9)
Simultaneous multiple axis motion	(see G98 or G99)

The CNC programmer can specify that multiple axes be moved simultaneously in a G-code move. The controller will then move all the specified axes to the specified targets. However, there are some subtle points concerning the velocities at which these moves will be accomplished.

A G0 command is a rapid move, while G1/G2/G3 are contoured moves. A G0 will move each axis at that axis' rapid feedrate (found under the Machine Parameters Screen), and no effort is made to coordinate the separate axis' motion. Each axis will finish its move at different times, and the G0 is not complete until the last axis has completed its motion.

G1, G2, G3, G12, and G13 are contoured moves. The axes will move in a coordinated fashion so that all axes finish their motion at the same time. The speed of the motion is determined by the E and F words (found under the CNC Parameters Screen). However, the way in which these speeds are determined from the E and F word can be complex in some cases, especially if the user moves rotary and linear axes simultaneously. Refer to Chapter 1, Section 1.2.2. Linear Feedrate.

G1 is a linear move, while G2 and G3 are circular moves. The user can specify a linear and circular move to happen simultaneously by putting a G1 and a G2/G3 on the same line. Also, the user can perform two circular moves simultaneously by using G12 and G13. These G-codes are the same as their "-10" counterparts for simultaneous motion. For example, the following code performs two circular moves simultaneously.

```
G2 I1 J1 X0 Y0 G13 I3 J3 Z0 A0
```

All CNC G code motion statements will wait until their motion is complete before proceeding onto the next CNC statement. These CNC statements are called synchronous. The controller does provide asynchronous CNC motion statements that initiate a motion, but then immediately continue on to the next CNC statement (see INDEX or STARTM extended commands). If such a statement is executed, a following CNC statement is executed that tries to move an axis still moving due to the first asynchronous motion statement, then that statement will wait until the first motion is complete before executing.

If an axis to be moved in a contoured or rapid move is rotary, then the controller will move to the target in the direction in which the least distance is traveled. For example, if B is rotary and at 0 degrees, then a move to 270 degrees will cause a counterclockwise rotation of 90 degrees, Not a clockwise rotation of 270 degrees. If the amount to be moved is exactly 180 degrees, then the motion will be clockwise.

There are some minor considerations due to the fact that the user specifies floating point numbers, but the actual positions are integers (counts). Normally, these are insignificant because the errors are always less than a count. However, it does mean that the floating point numbers reported as positions/velocities will not exactly match the floating point values specified by the user. The following information is provided for those wanting to know the details of the floating point to counts conversion. First of all, the controller will always truncate position values. This means that the move may be as much as one count short of the desired position. For example, if there are 100 counts per foot, and the user specifies a move of 1 inch, the controller will move 8 counts, not 8.333 counts. Velocities will also be truncated to the nearest "user-unit" (user-units can be either mm or inches) per second. Due to truncation, the controller may not be able to satisfy the acceleration, the truncated velocities, and the distance exactly, since the move velocity and distance were computed correctly in floating point. What the controller does, is satisfy both the truncated distance and velocity exactly and make up for any mismatch in the decel phase of the move. Therefore, the slope of the velocity during decel may vary slightly from what is expected. A final consideration is for profiled or blended moves, (see section 2.4.7.) that do not have a decel phase. Here the velocity during the constant phases is reduced, so that the truncated distance is satisfied.

2.3. Motion G-codes

2.3.1. Point-to-point Positioning at a Rapid Feedrate (Motion) **G0**

The G0 command specifies axis movement for unsynchronized point-to-point movement at the rapid transverse feedrate (machine parameter "Rapid Feedrate"). Acceleration and deceleration ramp time and type are specified by the axis parameters ACCEL, DECEL, ACCELMODE, and DECELMODE. *Reference G10 command for same on Axis Plane 2.*

Typically, moves of this type are used for such operations as moving a tool to the workpiece or moving a finished part out to the unloader. To keep machining time to a minimum, these moves need to be performed as quickly as possible.

It should be noted that although multi-axis moves of this type begin at the same time, all axes may not finish at the same time (each axis may have a different rapid traverse feedrate associated with it, along with a different target displacement).

Refer to the *UNIDEX 631/U600 User's Manual* for a detailed description of the rapid feedrate machine parameter.

SYNTAX: *G0 axis name and move distance [axis name and move distance]*

EXAMPLE

G0 G90 X10.	;Moves to X10. using rapid traverse feedrate
G0 G91 X5. Y10.	;X5.0 and Y10.0 using rapid traverse feedrate

The G0 feedrate is limited to 100% MFO maximum.



2.3.2. Linear Interpolation (Motion)**G1**

The G1 command specifies synchronized linear contouring at the commanded vectorial feedrate (F). This differs from a G0 type move in that all axes commanded to move begin and end at the same time. The resulting motion is a straight line from the current position to the position specified in the parts program block. *Refer to G-code summary and F word documentation for motion details.*

SYNTAX: *G1***EXAMPLE:**

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



The acceleration/deceleration type used is defined by the current operational mode of the Ramp Type and Accel Mode G-code groups.

2.3.3. Circular Interpolation CW (Motion)**G2**

The G2 command causes an arc to be generated by the coordinated motion of two axes. When viewing the axes' plane from the negative direction of a perpendicular axis (per the right hand rule), the arc direction is clockwise (CW). *Refer to G-code summary for motion details.*

The arc is assumed to start at the current position and end at the position specified in the parts program block. You must describe the radius of the arc by specifying the offsets to the centerpoint of the circle. This is done using the I/J/K keywords, where I/J/K are associated with the X/Y/Z plane axes as specified in the G17 code respectively. The offsets are always measured relative to the current position, regardless of the G90/G91 setting. The order in which X, Y, I, and J appear on the line is irrelevant. The controller checks the current ending and center points for validity. The distance between the center and current position must be equal to the distance between the ending position and the center point (within 15%). If this condition is not true, the user will get a "Radius Error" CNC fault.

The axes' which are affected by the G2 command are those associated with the axes' plane currently active (Plane Select G-code group, G17/G18/G19). The vectorial feedrate at which the motion is to occur is specified using the F keyword. If the ending position is equal to the current position, a 360° circle will be executed.

SYNTAX: *G2 AxisName/EndPt AxisName/EndPt IJK/CenterPt IJK/CenterPt*

The CNC supplies a default value of zero for any parameters omitted from the program block (such as an End Point or Center Point). The minimum set of parameters which must be supplied is either the X or Y parameter and the I or J. Therefore,

G2 X1.0 Y1.0 I1.0	==	G2 X1.0 Y1.0 I1.0 J0.0
G2 X2.0 I1.0	==	G2 X2.0 Y0.0 I1.0 J0.0
G2 I1.0	==	G2 X0.0 Y0.0 I1.0 J0.0



To accomplish helical interpolation it is necessary to program two axes to do circular interpolation and a third axis to do linear interpolation. This is accomplished by specifying a G2, G3 and G1 on the same line. Also, the user can specify two circles to be executed simultaneously via G12 or G13.

EXAMPLE:

G90 G2 X.5 Y3 I0 J-2	;Starting coordinate = {.5,3}
	;Ending coordinate = {2.5,1}
	;Center coordinate = {.5,1}

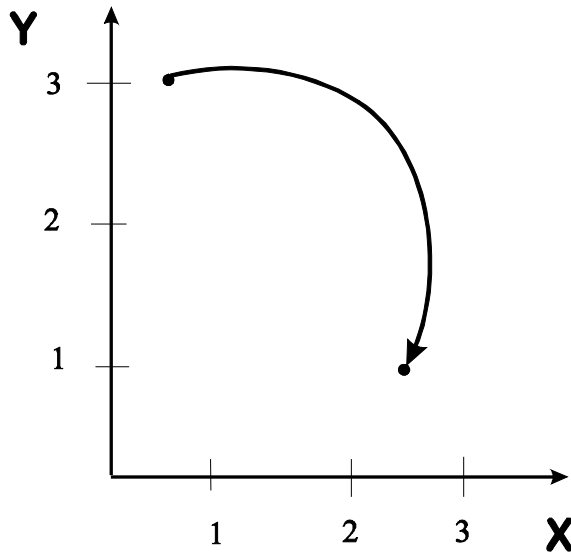


Figure 2-1. CW Circular Interpolation

2.3.4. Circular Interpolation CCW (Motion)

G3

A G3 generates a counterclockwise (CCW) arc and in every respect is identical to a G2 command. Compare Figure 2-2 with Figure 2-1 under G2.

SYNTAX: G3 AxisName/EndPt AxisName/EndPt IJK/CenterPt IJK/CenterPt

EXAMPLE:

G90 G2 X.5 Y3 I0 J-2	;Starting coordinate = {.5,3}
	;Ending coordinate = {2.5,1}
	;Center coordinate = {.5,1}

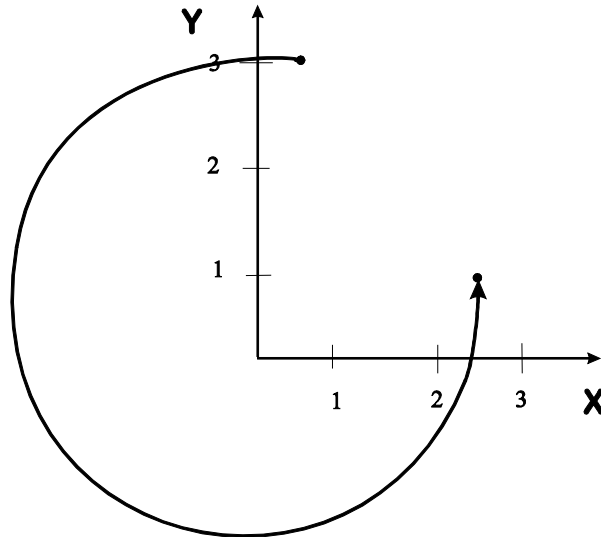


Figure 2-2. CCW Circular Interpolation

2.3.5. Dwell**G4**

This command causes a delay in program execution. The duration of the delay must be specified in seconds using the F keyword. The resolution of the timer used is 0.001 seconds or 1 millisecond.

SYNTAX: *G4 Fnn* (where nn is the delay given in seconds)
 G4 \$var (where the variable named *var* specifies the dwell time in seconds)

EXAMPLE:

G4 F5.	;Dwell 5 seconds
--------	------------------



The dwell command must occupy its own block within a program.

The value of the modal F keyword (vectorial feedrate) is unaffected by this command.

2.4. G8 and G9 Overview

G8 and G9 relate to how a controller behaves when linking two CNC moves together. The controller, by default, will blend two moves together smoothly accelerating to the new speed between the two moves. G8 and G9 can be used to alter this behavior. G9 forces the controller to decelerate to zero smoothly at the end of the block, the G9 appears. G8 forces the acceleration and deceleration to the new speed to be instantaneous.

G8 and G9 can be used separately or together, see example below. However, the G8 and G9 settings only apply to the block that they appear.

G8 and G9 have no effect on G0 moves, the controller always decelerates smoothly to 0 in between G0 moves.



EXAMPLE:

```
G91 G68.                ; Relative coordinates, linear accel/decel
G1 X10 F10.
G1 X10 F10 G9.
G1 X10 F10 G8.
G1 X10 F10 G8 G9.
```

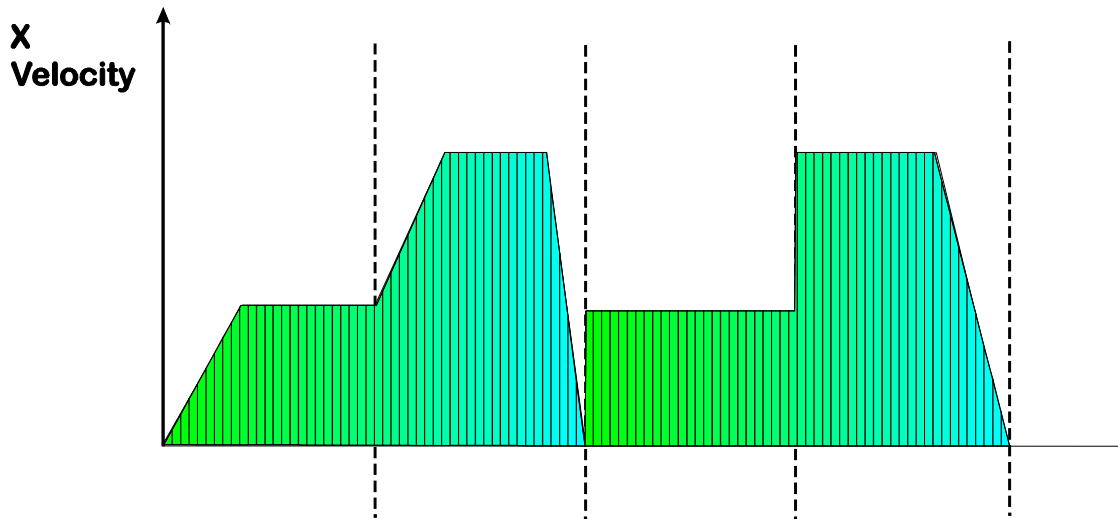


Figure 2-3. G8 and G9 Velocity Profile

2.4.1. Instantaneous Acceleration

G8

This command causes the axis to accelerate to the new velocity instantaneously (Figure 2-4 shown below, displays the velocity profile with G8). Without this command, acceleration is performed based upon the current settings of the Accel Mode and Ramp Type operational modes.

SYNTAX: G8

EXAMPLE:

```
G90 G1 G8 X1. F100.
G90 G1 X1 F100.
```

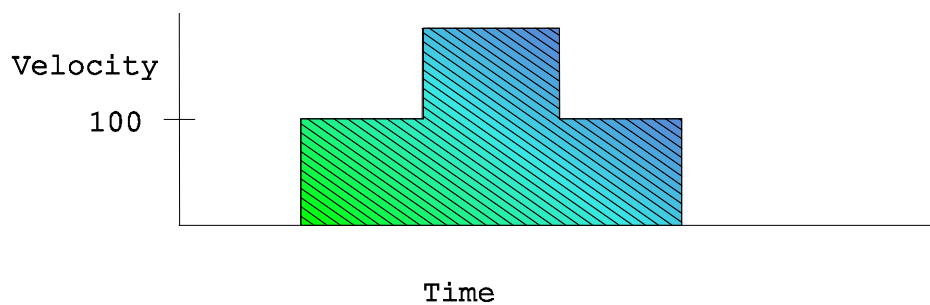


Figure 2-4. Velocity Profile With G8



Forced instantaneous acceleration, even when a G8 is not active can occur under some circumstances.

If a move in G8 mode is feedheld, the deceleration to zero is instantaneous. The same applies to MFO adjustments during a G8 mode move.

Corners are the most common case. Examine the following code fragment, where X and Z are linear axes.

```
G90 G0 X0 Z0          ; Goto {0,0}; use absolute coordinates from now on
G1 X100 Z0 F100       ; X at 100 units/sec ; Z does not move.
G1 X100 Z100 F141.4   ; X at 100 units/sec ; Z at 100 units/sec
```

The controller will not decelerate in between the two moves (no G9 specified); so the X axis will be traveling at 100 unit/min. at the instant the second move begins. This means that the Z axis must also be 100 units/min. at that instant, in order for the two motions to finish at the same time. In the last instant (the end of the first move) the Z axis was not moving.

This results in an instantaneous velocity change (infinite acceleration) of the velocity command of the Z axis.

There are only two sensible alternatives here.

1. Slow down to a stop in between the moves (this violates the speed behavior specified by the user).
2. Ramp the Z axis up as quickly as possible starting at the beginning of the second move (this violates the position contour specified by the user).

It is by no means clear which of these solutions is preferable, it depends on the application. So the U31 controller leaves it up to the user to decide. If solution 1 is desired, the user must place a G9 in the first G1 block shown above. Solution 2 can be accomplished with an axis parameter especially provided for this situation, VELTIMECONST. When this constant is non-zero, a low-pass filter (removes high frequency components) is applied to the velocity command. The constant is approximately proportional to the number of milliseconds that the filter will spread out an instantaneous velocity change. It is strongly recommended that if this is to be used, that it is tested under benign conditions first, in order to insure the proper setting for this constant. If neither of these solutions are applied here, then the velocity command will have an instantaneous jump in it. However, it is clear that the actual velocity will never be instantaneous and will be spread out by some amount based on the actual mechanics being used. If neither a G9 or the VELTIMECONST is used by the programmer, then the mechanics will impose a form of solution 2, or the axis will throw a current or position fault.

There can also be instantaneous decelerations, for similar reasons. Examine the following example:

```
G90 G0 X0 Z0 ; Goto {0,0}; use absolute coordinates from now on
G1 X100 F100 ; X at 100 units/sec
G1 Z100 F100 ; Z at 100 units/sec
```

In this example the X axis will decelerate to 0 instantaneously at the end of the first block, while simultaneously the Z axis accelerates instantaneously. However, note the following exception, if the Z axis is rotary instead of linear (and we are linear dominant), then the Z axis will accelerate smoothly at the beginning of the second move. Because rotary axis speeds are handled separately. Since the first block has no rotary component, and the second block no linear component, the the rotary axis is not constrained and can accelerate normally.

Finally, another case in which instant deceleration can occur is after a sequence of blended moves where there is no deceleration at the end of each move, except for the last move in the sequence. If the last move specifies a very short distance, relative to the velocity, then the controller does not have time to follow the specified deceleration. It must apply a deceleration which brings it to the specified target from the speed achieved in the last move.

This can result in decelerations much faster than specified by the user and can generate instantaneous decelerations.

2.4.2. Force Deceleration**G9**

The G9 command forces the axes to decelerate to zero velocity at completion of the move (Figures 2-5 and 2-6 give a comparison of the velocity profile with and without G9). The subsequent move will then accelerate from zero velocity to the commanded feedrate. The following example illustrates the effect of this command upon a sequence of motion blocks.

SYNTAX: G9

EXAMPLE: Velocity profile without G9

```
G91.
F60.                ;Set the vector feedrate to 60
G1 X1.             ;Move the X axis 1.0
F120.              ;Set the vector feedrate to 120
G1 X2.             ;a second time 2.0
F60.               ;Set the vector feedrate to 60
G1 X1.             ;and another move of 1.0
G4 F1.             ; Dwell for 1 second
G1 X1.             ;This move will have an acell, because a G4 preceded it
```

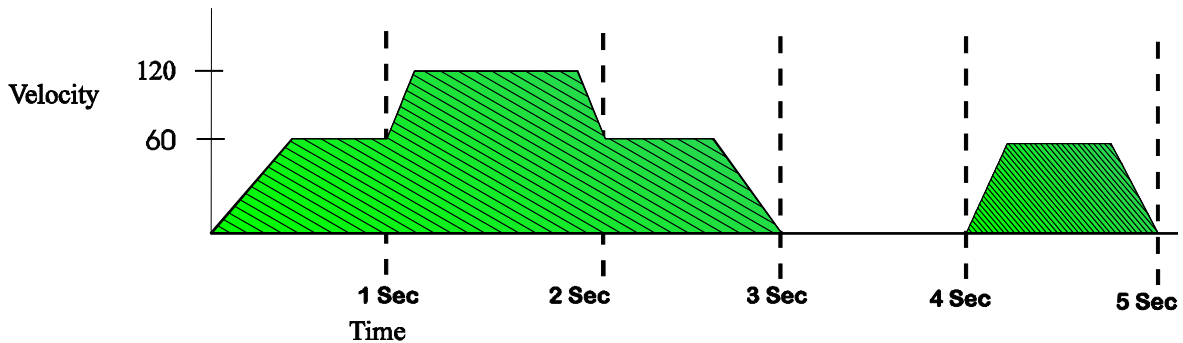


Figure 2-5. Velocity Profile without G9

EXAMPLE: Velocity profile with G9

```
G91.
F60.           ;Set the vector feedrate to 60
G1 G9 X1.     ;Move the X axis 1.0
F120.        ;Move at a Velocity of 120
G1 G9 X2.     ;a second time 2.0
F60.         ;Set the vector feedrate to 60
G1 G9 X1.     ;and another move of 1.0
G4 F1.       ;Dwell for 1 second
```

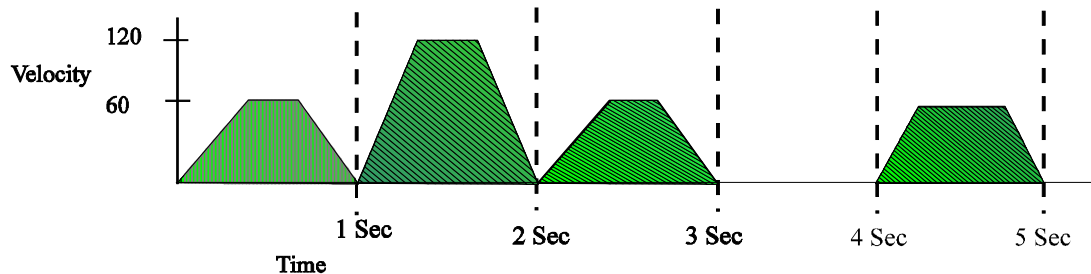


Figure 2-6. Velocity Profile with G9

The user should note that when executing a return move (from jog and return) that the controller will always decelerate to zero before resuming the move (if any) that was interrupted by the jog and return.

Even if the user does not place a G9 in the line, there are some exceptions in which the controller will force a deceleration at the end of the move (and a subsequent acceleration at the beginning of the next move).

An exception is when the user is running in step mode, in this case the controller decelerates at the end of all motion blocks. Another exception is when certain CNC statements lie in between the two moves to be blended (such as a G4 in the example above). If the intervening statement causes a motion stop, or specifies a synchronous action with another processor, the controller will force a deceleration. A list of these statements are below.

Other motion codes - G4, G0

Program control M codes - M00, M01, M02, M30, M47

Any CDW operation - OPENCDW, CLOSECDW, DISPLAY, RECORDON,...

Any FILE I/O operation - FILEREAD, FILEWRITE, FILERESET, FILECLOSE,...

Any DATA collection - DATA

Any Wait operation - WAIT, WTCH

Any statement containing a conditional - IF, WHILE, RPT, JUMP (with conditional clause)



The setting of the BLOCK DELETE has no effect on this, the controller sees the intervening statements regardless of whether they are inactivated by block delete or not.

A third exception is when normalcy is active (see Section 2.4.1) and the controller is on a corner between two G-codes executing a normalcy move. The controller will decelerate before, and accelerate after the normalcy move regardless of the G9 setting.

2.4.3. Circular Interpolation CW (Motion)

G12

The G12 command causes an arc to be generated by the coordinated motion of two axes. This command is identical to a G2, except that the axis plane is set through G27, G28, and G29 instead of G17, G18, and G19. The G12 provides the programmer the ability to perform two circular motions simultaneously. This is accomplished by putting a G12 or G13 on the same line as a G2 or G3.

SYNTAX: *G12 AxisName/EndPt AxisName/EndPt IJK/CenterPt IJK/CenterPt*

EXAMPLE:

G90 G11 X2.0 Y-0.5 I1.0 J-0.1	;A CW arc will be produced from the current ;position to the (2,-0.5) coordinate position, ;the center point being (1.0, -0.1). The I and ;J are incremental offsets from the starting ;position and are independent of G90 or G91.
-------------------------------	---

2.4.4. Circular Interpolation CCW on Axis Plane 2

G13

The G13 command causes an arc to be generated by the coordinated motion of two axes. This command is identical to a G2, except that the axis plane is set through G27, G28, and G29 instead of G17, G18, and G19. The G13 provides the programmer the ability to perform two circular motions simultaneously. This is accomplished by putting a G13 or G12 on the same line as a G2 or G3.

This command is identical to the G3 command.

SYNTAX: *G13 AxisName/EndPt AxisName/EndPt IJK/CenterPt IJK/CenterPt*

EXAMPLE:

G90 G13 X-2.0 Y-0.5 I-1.0 J-0.1	;A CW arc will be produced from the current ;position to the (-2, -0.5) coordinate position, ;the center point being (-1.0, -0.1)
---------------------------------	---

2.4.5. Spline Move G30

The G30 command provides you with an additional method of generating contoured motion. When using this method, you specify a set of points which must be approximately a fixed vectorial distance apart. The UNIDEX 631/U600 CNC will then use a splining algorithm, which provides a smooth transition from point to point, to determine the desired path.

The following illustration, Figure 2-7, demonstrates the effect of the splining algorithm on a specific set of points.

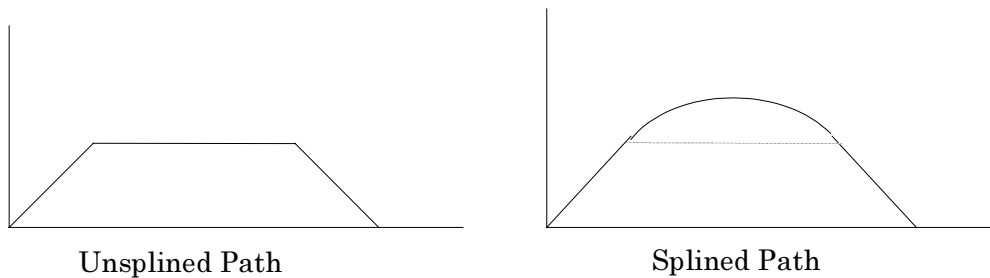


Figure 2-7. Effect of Splining Algorithm

To use the splining feature of the UNIDEX 631/U600 CNC, you must first command the CNC to enter the splining mode. This is done with the G30 command. The points along the desired path are then listed individually, in the current programming mode either absolute (G90) or incremental (G91). The vectorial distance between these points should be approximately the same. Another command within the same G-code group (G0/G1/G2/G3) is used to terminate this operational mode.

When the CNC encounters the G30 command within an executing parts program, program execution will pause while the spline coefficients are calculated. When the calculations are completed, program execution will resume and the axes will move to the specified points. The speed at which this motion occurs is specified using the feedrate (F) keyword and applies to all points specified. Feedrate override controls are active while executing splined moves.

SYNTAX: G30

EXAMPLE:

```
G30 ;Enter the cubic spline mode
F100. ;Specify a feedrate for the entire splined path
G91 X1. Y1. ;First point (1,1)
G91 X2. ;Second point (3,1)
G91 X1. Y-1. ;Third point (4,0)
G1 ;Exit the cubic spline mode
```

These points may be used to generate the profile shown in Figure 2-7.



This is not the default operational mode of the CNC.

There must be a minimum of three G30 move blocks or the coefficient calculation fails.

2.4.6. Constant Lead Thread Cutting

G33

This term refers to an automated motion cycle that is designed to place threads into an outer edge of a part. In this context, the term "threads" is used as it would be when referring to the "threads" of a bolt.

MODAL

In order to utilize this feature, the part must be mounted onto the spindle axis. The cutting tool is under the control of two other axes, specified as ThreadX and ThreadY on the CNC Initialization screen. (Refer to the *UNIDEX 631/U600 User's Manual*.) The ThreadX axis is perpendicular to the center line of the spindle and the ThreadY axis is parallel to the centerline of the spindle. See Figure 2-8 below.

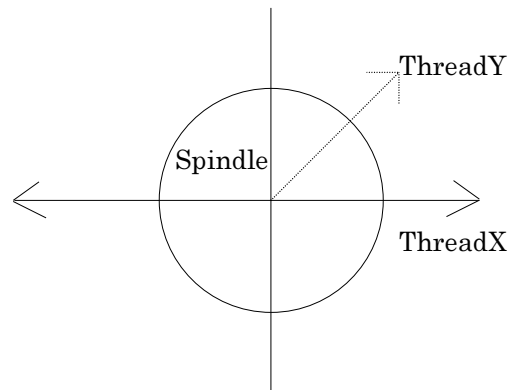


Figure 2-8. Threading Axes Designations

When initiating a threading cycle, you must specify several parameters. These include the distance between threads (lead) and the length of the part to be threaded. The thread lead is specified in user units (in/mm) and must be greater than zero. The thread length is also specified in user units (in/mm) and may be specified in either absolute coordinates or incremental distances, dependent upon the current operational mode (G90 or G91).

Linear or tapered threads may be cut on the part. To accommodate tapered threads, the G33 command provides an optional parameter, Thread Taper, that is used to specify the angle of the part's outside surface with respect to the centerline of the part. This angle is specified in degrees with a valid range of -45° to 45° . If this parameter is not specified, the Thread Taper angle is assumed to be zero. Refer to Figure 2-9.

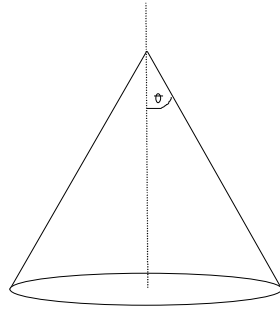


Figure 2-9. Thread Taper Angle

You also have the capability of specifying the rotation angle at which the threads are to begin. This angle is also specified in degrees and has a valid range of 0° to 359° . If omitted, the starting angle is assumed to be zero. A taper must be specified in order to specify the rotation angle. Refer to Figure 2-10 below.

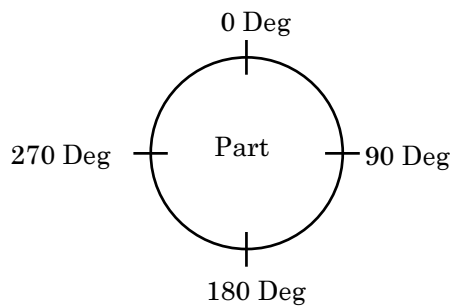


Figure 2-10. Threading Start Angle

The last parameter of the G33 command is also optional. If used, it specifies the distance which the ThreadX axis should retract upon detecting a feedhold condition. This distance is specified in user units (in/mm) and is always considered to be incremental. The sign of this value is used to denote directional information. If this parameter is omitted, a retract distance of zero is assumed. A taper and starting angle must be specified if the retract distance is specified.

The retract position is activated when a thread cutting cycle is being executed and feedhold is activated. The axes will make a linear interpolated perpendicular move to the retract position. When the safe retract position is reached, axis motion stops and the system is in a feedhold. The operator may then resume thread cutting by pressing the Cycle Start control (refer to the CNC Run screen in the *UNIDEX 631/U600 User's Manual*).

The thread cutting cycle itself uses the current spindle speed and thread lead to calculate the feedrate at which the linear axes are to move. If the linear velocity is not attainable, an error will appear on the display and the cycle will be aborted.

While a thread cutting cycle is in progress, the spindle speed override (MSO) may not be used to increase the spindle velocity. It will, however, permit you to decrease the speed.

SYNTAX: *G33 ThreadY/Lead Length [TaperAngle] [StartAngle] [RetractDist]*

EXAMPLE:

G33 Y0.125 3.0 30 0 -25.	;Begin threading cycle. Thread lead is 0.125, ;length of the thread is 3.0, thread taper is ;30° and the starting part orientation is at 0°. ;If a feedhold is activated, the axis will retract ;25.0 in the negative direction.
--------------------------	--

G33 X[lead] [length] <taper> <sync> <retract>



G33 is not functional.

2.5. Plane Selection Codes

The commands contained within the Plane Select G-code group describe the set of axes upon which circular interpolation is to be performed. This information is necessary when attempting to default missing parameters to circular interpolation commands (G2 and G3, Set #1, G12 and G13, Set #2).

2.5.1. Plane Selection Codes Set # 1

G17/G18/G19

As mentioned in the discussion of the I/J/K keywords, the CNC Parameters Plane Selection Menu assigns three axis pairs on which circular interpolation may be performed. These three axes are the Plane_X_Axis, the Plane_Y_Axis and the Plane_Z_Axis. The G-codes found in the Plane Select group are used to select one of these pairs.

The following table (Table 2-2) defines the axis planes applicable to each of the G-codes found in the Plane Select group.

Table 2-2. Axis Planes for G-codes in Plane Select Group

G-code	Axis Plane Selection
G17	Plane_X_Axis and Plane_Y_Axis
G18	Plane_Z_Axis and Plane_X_Axis
G19	Plane_Y_Axis and Plane_Z_Axis

SYNTAX:

G17
G18
G19

EXAMPLE:

G17 G2 X1.0 Y1.0 I1.0 J0.0	;Make arc with X and Y axes (1st axes plane 1)
G18 G2 X1.0 Z1.0 I0.0 K1.0	;Make arc with Z and X axes (2nd axes plane 1)
G19 G3 Y-1.0 Z-1.0 J0.0 K-1.0	;Make arc with Y and Z axes (3rd axes plane 1)

In order to fully understand the need for axes planes, consider the following program block.

G1 G2 X1.0 Y1.0 Z1.0 I1.0

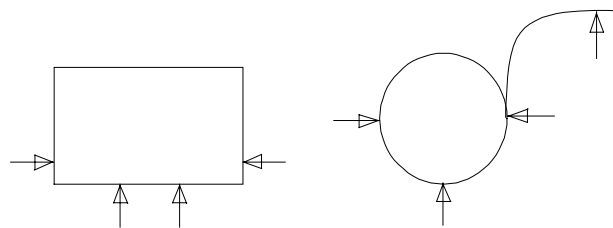
In this example, two axes are programmed to make an arc (of radius 1.0) and the third axis is to make a linear move of 1.0. Without an axes plane designation, it would be impossible to determine which two axes perform circular interpolation. However, by specifying the axis plane designation, the block can be uniquely decoded as follows.

Table 2-3. Decoding a Block of a Selected Axis Plane

Program Block	Circular Portion	Linear Portion
G17 G1 G2 X1.0 Y1.0 Z1.0 I1.0	G2 X1.0 Y1.0 I1.0 J0.0	G1 Z1.0
G18 G1 G2 X1.0 Y1.0 Z1.0 I1.0	G2 X1.0 Z1.0 I1.0 K0.0	G1 Y1.0

2.6. Normalcy Mode (G20, G21, G22)

Certain types of cutting tools require that they be oriented perpendicular (normal) to the part being cut. Such tools are typically mounted to a rotary axis so that the orientation of the tool may be changed as the position of the part changes. Figure 2-11 below shows the required orientation of such a cutting tool, with respect to the part being cut.



—▷ Denotes Tool Orientation

Solid line is the part

Figure 2-11. Tool Orientation

Although maintaining this perpendicularity is possible using conventional G-code programming, it would be very cumbersome. The moves would have to be broken up into small segments and extensive calculations would be required to calculate the appropriate rotary move distances and feedrates.

The UNIDEX 631/U600 CNC provides a feature which alleviates the parts programmer from this duty. The operational mode which provides this feature is referred to as the "Normalcy" mode of operation, since it is necessary to keep the tip of the tool normal to the surface of the part being cut.

In order to operate in this mode, several pieces of information must be supplied by the operator. This includes the axis to which the cutting tool is attached (B-Axis) and the axes which make up the plane to which normalcy must be maintained (XPlane/YPlane). A normalcy speed must also be provided (refer to next paragraph for details). All of these parameters may be modified within the B Axis Initialization dialog box. Please refer to the *UNIDEX 631/U600 User's Manual* for more information on the operation of this dialog box.

While operating in the normalcy mode, the CNC automatically maintains the relationship between the B-Axis and the plane defined with the XPlane/YPlane parameters.

Once the tool achieves normalcy during a G1 move, no normalcy movement is required for the remainder of the move. However, at the start of a G1 move, normalcy movement of the axis prior to the move may be required. For example, following the path of the rectangle shown in Figure 2-11, the normalcy axis must rotate 90 degrees at each corner before proceeding down the next side. This is called a "Normalcy Alignment Move" that is accomplished with the G0 command and moves at the rapid traverse axis speed (see Machine Parameters Screen). The speed of this move is not limited by the normalcy speed limit.

G0 motion does not appear on the feedrates screen under the actuals column.



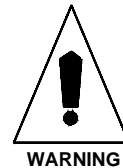
It is important to realize that the normalcy alignment move will force the previous motion block to decelerate to 0 prior to the normalcy move. This is true even if there is no G9 on the first block. Also, there are some conventions concerning the direction of rotation of the normalcy alignment move; these are discussed under G21 and G22.

Unlike G1 moves, circular moves (G2 and G3) require the normalcy axis move continuously throughout the move. The speed that the normalcy axis must move is determined by the radius of the circle and feedrate along the circular path. This is much like a complex move where the rotary and linear axes move and the linear axis is dominant (refer to Feedrate and Spindle Speed Codes). Meaning, the speed and acceleration of the rotary axis is slaved to the speed and acceleration of the linear axis. Therefore, much of the same problems can occur here, as in a complex linear dominant move. The G98 overview fully discusses these problems, this section just mentions how they differ from the cases discussed under the G98 command.

- Rotary axis Feedrate limiting
- Rotary axis Feedrate faults
- Rotary axis accelerations/decelerations.

Rotary axis feedrate limiting occurs if the normalcy axis travels too fast during a circular move. The speed limit applied is the normalcy speed specified by the user.

In normalcy moves, the speed of the normalcy axis move is not limited by the rapid feedrate or the E word setting. It is only limited by the normalcy speed specified under the B axis screen, under the CNC parameters menu item.



Just as in complex moves, situations can occur where the controller cannot properly limit the normalcy axis speed. In these cases a fault message will be generated, “Normalcy Speed Exceeded”.

The normalcy axis acceleration and deceleration will be forced based on the acceleration and deceleration of the linear axis. No acceleration/deceleration limits are applied.

Normalcy has no effect on the G0 moves.



The rotary axis cannot be moved explicitly with G1/G2/G3 while normalcy mode is in (G21/G22).



2.6.1. Disable Normalcy Mode**G20**

This command disables the mode of operation in which the cutting tool is automatically kept perpendicular to the part being cut (normalcy mode).

Please refer to the Normalcy Mode Overview for a general description of the implementation of this feature on the UNIDEX 631/U600 CNC.

SYNTAX: *G20*

EXAMPLE:

G20	;Disable the normalcy mode
-----	----------------------------

This is the default operational mode of the controller.

MODAL**2.6.2. Activate Normalcy Mode Left****G21**

This command activates the mode of operation in which the cutting tool is automatically kept perpendicular to the part being cut (normalcy mode). In this mode the normalcy alignment move will always occur in a clockwise direction. This may lead to unexpected results on inside corners, refer to Figure 2-12.

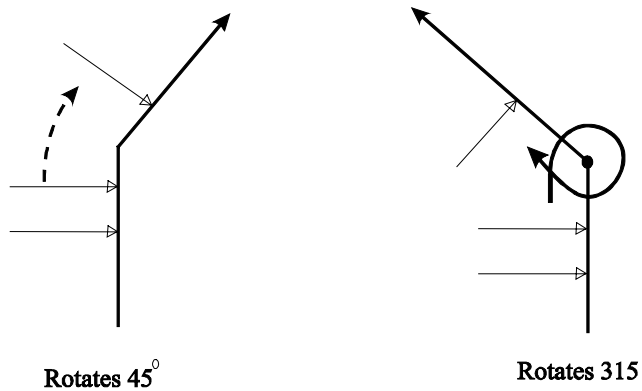
MODAL

Figure 2-12. Normalcy Left

Please refer to the Normalcy Mode Overview for a general description of the implementation of this feature on the UNIDEX 631/U600 CNC.

SYNTAX: *G21*

EXAMPLE:

G21	;Enable normalcy mode to the left of the part
-----	---

2.6.3. Activate Normalcy Mode Right

G22

This command activates the mode of operation in which the cutting tool is automatically kept perpendicular to the part being cut (normalcy mode). In this mode the normalcy alignment move will always be performed in a counterclockwise direction. This may lead to unexpected results on “inside corners”, refer to Figure 2-13.

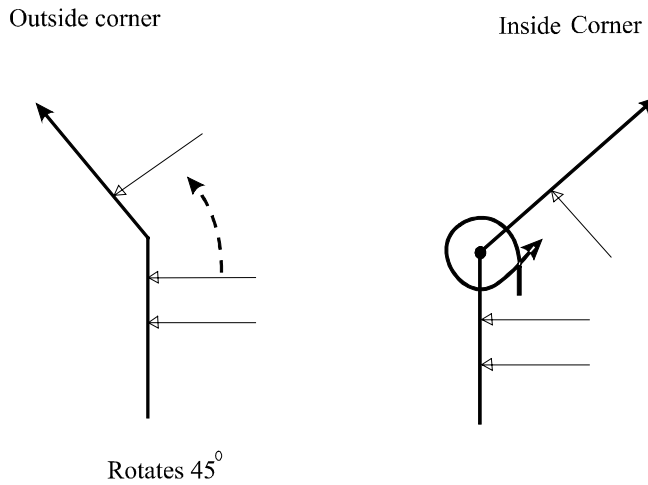


Figure 2-13. Normalcy Right

Please refer to the Normalcy Mode Overview for a general description of the implementation of this feature on the UNIDEX 631/U600 CNC.

SYNTAX: G22

EXAMPLE:

G22	;Enable normalcy mode to the right of the part
-----	--

2.7. Safe Zones (G36, G37)

In some applications, it is desirable to define a particular area in which all axes' motion may occur. Conversely, other applications require the ability to define an area in which axes are not permitted to enter. The UNIDEX 631/U600 CNC "Safe Zone" feature provides you the ability to perform either of these two functions.

As described in the *UNIDEX 631/U600 User's Manual*, the implementation of safe zones in the UNIDEX 631/U600 CNC requires that you specify three parameters for each axis. The first, the "SafeZoneCW" parameter specifies the clockwise boundary of the safe zone. Similarly, the "SafeZoneCCW" parameter specifies the counter-clockwise boundary. The "SafeZoneMode" parameter specifies the type of safe zone being used (never enter -vs. never leave).

2.7.1. Enable Safe Zones

G36



The G36 command enables a safe zone, as well as specifies all applicable parameters. Once enabled, safe zone checking is performed prior to the execution of each motion command. A safe zone fault occurs if you attempt to command motion which violates the safe zone restrictions. For more information on UNIDEX 631/U600 CNC fault handling, please refer to the *UNIDEX 631/U600 User's Manual*.

The parameters for this command include a list of axes for which safe zones are to be activated, as well as the CW and CCW boundaries of those safe zones. The mode of operation for these safe zones is specified using the P keyword. If this keyword is specified as "1", the safe zone parameters describe a one dimensional area in which axis motion is not permitted to enter. If P is specified as "2", the parameters describe a one dimensional area in which the axes are not permitted to exit. (All other values for the P keyword are invalid.)

SYNTAX: *G36 AxisName SafeZoneCW SafeZoneCCW Px*

EXAMPLE:

G36 X10. 20. Y5. 10. P1	;Enable a safe zone in the area of X(10,20) ;and Y(5,10) in which these axes are not ;permitted to enter
G36 X-100. X100. Y10. Y20. P2	;Enable a safe zone in which the X axis is not ;permitted outside the area defined as (- ;100,100) and the Y axis is not allowed ;outside (10,20).



Although multiple axes may be used in the specification of the safe zone for the P1 and P2 types, each axis is evaluated individually. See Figure 2-14.

Therefore, if a P1 safe zone is used to specify a multi-dimensional area in which a set of axes may not travel, you may not command motion "through" that area (refer to Figure 2-14).

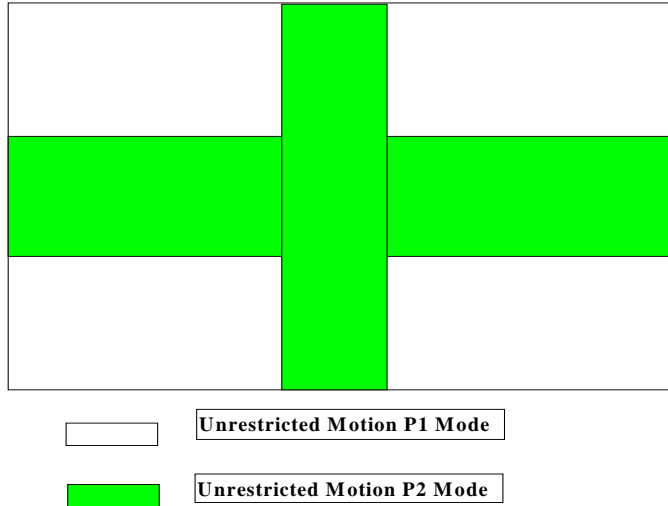


Figure 2-14. Unrestricted Safe Zones

Safe zone boundaries are evaluated only when computing the target position of any motion command. Therefore, if an axis is within the restricted area when a safe zone is enabled, a safe zone fault is not generated until the first motion is commanded. Furthermore, if the target position of that first motion command is outside the restricted area, the move will execute normally.

In order for the safe zone feature to operate as described, the SafeZone fault must be enabled for that axis. Refer to the *UNIDEX 631/U600 User's Manual* for more information on fault handling.

The safe zone parameters are interpreted as relative displacements if the CNC is in the G90 mode when the G36 command is executed.

2.7.2. Disable Safe Zones

G37

The G37 command disables safe zones which are currently active in the system. As with the enable safe zones command (G36), this command accepts parameters to permit you to specify the axes to which this command applies. Therefore, safe zones can be disabled individually or in groups.



Although no information, other than the axis designation, is needed to disable the safe zone, the command syntax requires that you supply a positional coordinate for each axis specified. This information is discarded by the CNC and should be programmed as zero.

SYNTAX: *G37 AxisName 0 [AxisName 0]*

EXAMPLE:

```
G37 X0. Y0. Z0. ;Disable the safe zones active for the X, Y and Z axes
```

The default operational mode of the CNC has safe zones disabled.



2.8. Intersectional Cutter Radius Compensation (ICRC) Overview

In cutting a workpiece, it is sometimes necessary to consider the radius of the cutting tool. For example, when an endmill cuts 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 tool's radius.

Intersectional Cutter Radius Compensation (ICRC) is an option that allows the operator to program the path along the outside edge of the cutter, without regard to the size of the tool. Without this option, the operator would have to offset the actual workpiece dimensions based on the radius of the tool.

This option significantly decreases the programming effort required for this type of application. It also allows the user to run the same program with tools of different diameters by simply changing the tool diameter information.

The user must provide a tool radius with the G43 command. The user must also provide the cutter compensation axes representing the plane where the compensation will be performed in the G44 command. The first axis in the G44 is called the horizontal axis, the second axis is called the vertical axis. Two circumstances of interest in cutter compensation are shown in Figure 2-15.

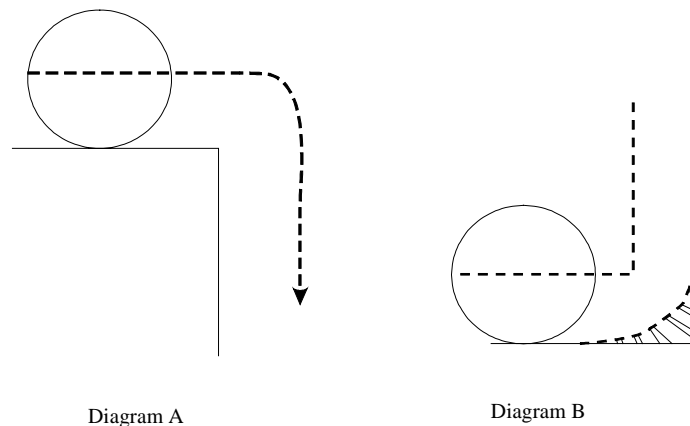


Figure 2-15. Cutter Radius Compensation Path

Diagram A in Figure 2-15 illustrates the controller will generate an additional 90 degree circular arc move, to move around the outside corner. The speed of this move will be the same as the last move (across the top edge) and there will be no deceleration between the user provided horizontal move and the auto-generated circular arc. If the user supplied the horizontal move with a G9, then the deceleration will occur during the circular arc. The speed along the circular arc will always be equal to the speed along the previous move. However, if the speed along the circular arc must be limited due to normalcy, then the previous move will be limited in the same way. Also, the circular move is not limited by the rapid feedrates. Diagram B in Figure 2-15 illustrates that for inside corners the controller will not generate an arc move, but the user should be aware that a small circular wedge of uncut material will remain in the inside corner.

When in cutter compensation mode, the PRESET positions will reflect the amount of the offset that the controller is applying to accomplish the compensation. For example, in Diagram A, during the first move, the PRESET position for the vertical axis will differ from the MACHINE position by an amount equal to the tool radius. The horizontal axis PRESET will be equal to the MACHINE position. As the tool moves around the circle, the offset will transfer itself to the horizontal axis. In the second move, the horizontal axis will show the offset while the vertical axis PRESET position will be equal to the MACHINE position.

In addition to the auto-generated arcs around the edges, tool compensation will automatically generate movement (lead-on and lead-off moves) entering (G41, G42) and exiting (G40) tool compensation mode.



The programmer can place statements in between two G-code motion lines, without affecting cutter compensation, as long as the intervening statement does not cause a pause in the movement. The same rules apply here for blending two moves without decelerating (see G9). If an intervening statement does break the motion (such as a M0), then the cutter compensation will pick back up once it locates a move after the intervening statement. This can have disastrous results if it is unintentional, as shown in Figure 2-16. Note that moves of axes not involving a cutter comp designated axis can be placed in between, with no effect.

When running in mirror mode (see G83), note that cutter compensation left becomes cutter compensation right, and visa versa.

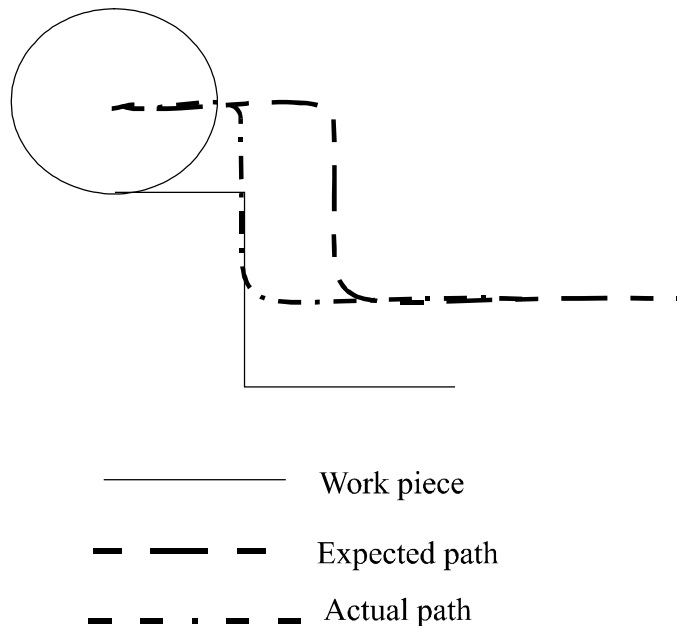


Figure 2-16. Cutter Compensation with Intervening Statements



2.8.1. Deactivate Cutter Compensation (ICRC)

G40

The G40 mode exits cutter compensation mode, and generates a lead-off move. How the lead off move is accomplished depends on whether the G40 is on its own line, or is on a line with a move. If the G40 is on a line with a G1 motion command, then a leadoff motion is accomplished within the G1 move, but if the G40 is on its own line, the controller simply sets the PRESET positions equal to the MACHINE positions - no movement is generated where the direction of the leadoff is perpendicular to the direction of the move. Refer to 1 and 2 in Figure 2-17.

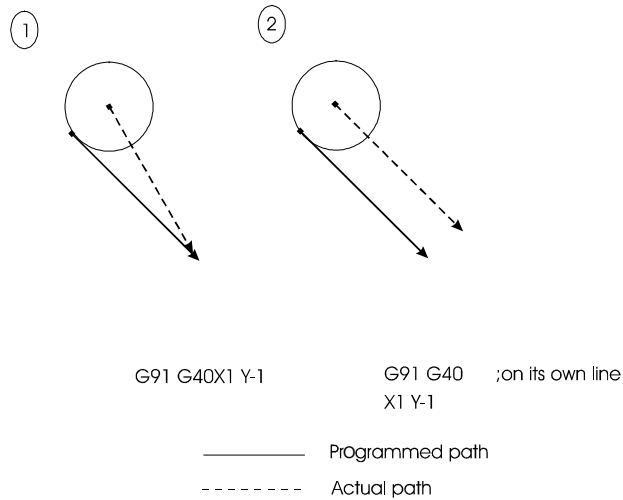


Figure 2-17. Lead Off Moves



If the user ends the program without a G40 (without a lead-off move) the program will throw a fault indicating this.

SYNTAX: *G40*

EXAMPLE:

```
G40 G1 X1. Y1. F100.           ;Deactivate the cutter radius compensation
                                ;and remove the offset during the end move
```

Please refer to the comprehensive example immediately following the G42 command.



The G40 mode is the default.

2.8.2. Activate ICRC Right

G41

G41 activates ICRC to the right of the programmed tool path relative to the direction of tool motion (see Figure 2-18). The center of the tool nose will then be kept on a line normal to the programmed path until ICRC is de-activated. The movement generated by a G41 differs based on whether it appears on its own line or on the same line as motion, refer to Figure 2-19.



An error is reported if the first motion block following ICRC activation commands circular interpolation on an axis that cutter compensation has been activated.

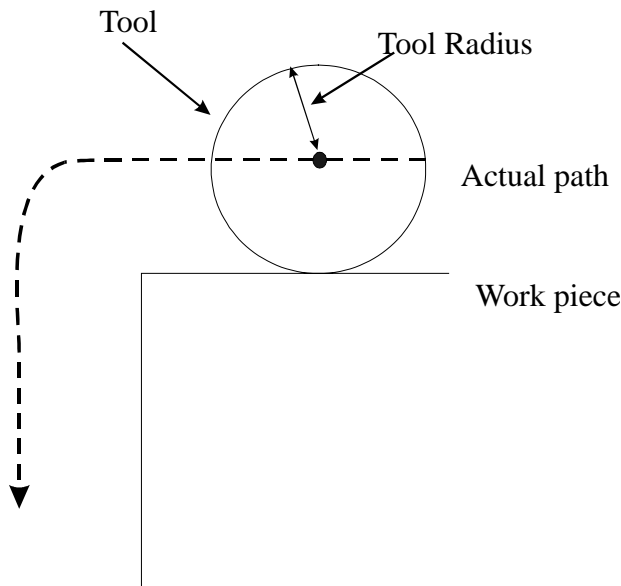


Figure 2-18. Path Compensation Right

Please refer to the Intersectional Cutter Radius Compensation Overview for a general description of the implementation of this feature on the UNIDEX 631/U600.

SYNTAX: *G41 [Lead-In Move]*

EXAMPLE:

G41 X2. Y2. F100.	;Activate the cutter radius compensation right
-------------------	--

Refer to the comprehensive example immediately following the G42 command.

The G40 mode is the default.



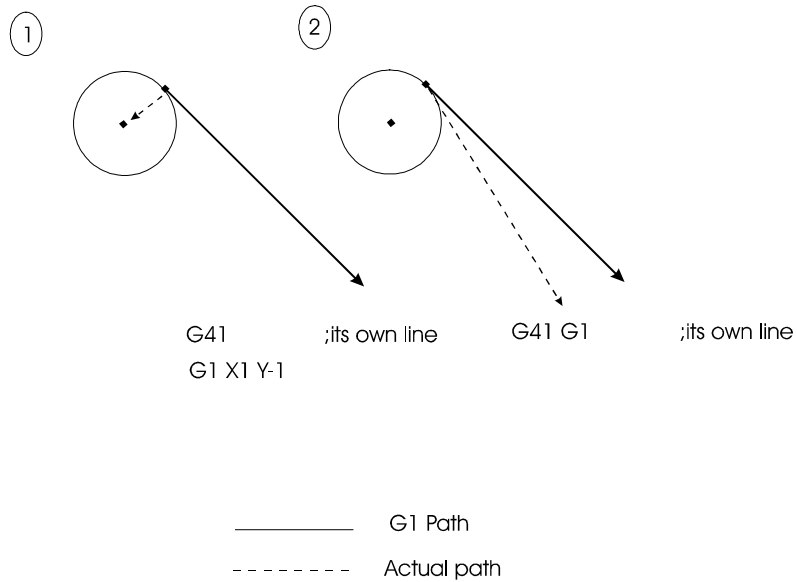


Figure 2-19. Lead-on Moves

A G41 will generate a lead-on move that moves the tool away from the piece by an amount equal to the tool radius. How this is done depends on whether the G41 appears on its own line, or on the same line as a G1 motion. If it appears on the same line as the G1 motion, then it will be applied during the motion and the direction of the lead-on move will be perpendicular (and to the right, looking in the direction of movement) to the direction of the executed G1 move.

If the G41 is on its own line, then the lead-off move is performed immediately by itself. The only drawback is, the direction of the lead-off move being made. The controller will make the lead-off move in the direction perpendicular (to the right, looking in the direction of movement) to the direction of the next G1 move. However, there are some circumstances where the controller cannot locate the next move. For example, a subroutine call, if statement, or M02 lies in-between the G41 and the next move. In these cases the G41 will not generate a lead-off move.



G42 is similar to G41 except it activates ICRC to the left of the programmed tool path.

2.8.3. Activate ICRC Left

G42

This command activates Intersectional Cutter Radius Compensation (ICRC) to the left of the programmed tool path relative to the direction of tool motion (see Figure 2-20). The tool offset will be incorporated into the execution of the next linear motion command. The center of the tool nose will then be kept on a line normal to the programmed path until ICRC is de-activated.



An error is reported if the first motion block following ICRC activation commands circular interpolation on either axes for which cutter compensation is being activated.

The behavior of G42 depends on whether it appears on its own line or on a line with motion, refer to the figure in margin to the left.

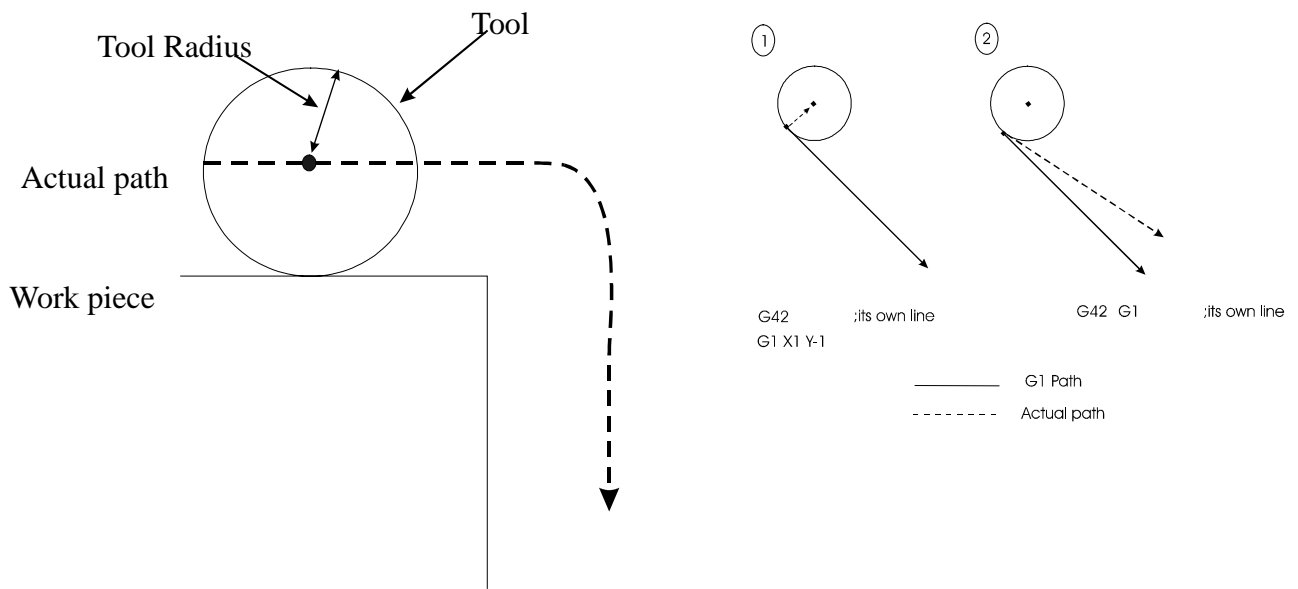


Figure 2-20. Path Compensation Right

Please refer to the Intersectional Cutter Radius Compensation Overview for a general description of the implementation of this feature on the UNIDEX 631/U600.

SYNTAX: *G42 [End Move]*

EXAMPLE:

```
G42 X2. Y2. F100. ;Activate the cutter radius compensation left
```

Please refer to the comprehensive example on the following page.

The G40 command is the default.



Cutter Compensation Example

Figure 2-21 demonstrates the effect of ICRC on the tool path. It assumes that the X and Y axes were designated for use with cutter compensation.

```

G18
G44 Z X          ; IMPORTANT: G44 must match the G17/G18 setting
G43 R0.5         ; set tool radius
G90 G0 Z0 X0    ; move to {0,0,0}
M0
G91 F100        ; Incremental mode positioning
G1 X.5 Z.5      ; move 1) Move with ICRC disabled.
G42             ; move 2) Enable ICRC left, and do lead-on move
G1 X1 Z1        ; move 3) Move while in ICRC left mode
G40             ; Disable ICRC without a lead-off move
G1 X.5 Z-.5     ; move 4) Do lead-off manually
G41 G1 X1 Z1    ; move 5) Enable ICRC right, and lead-on while doing a move
G1 X.5 Z.5     ; move 6) Move while in ICRC right mode
G40 G1 X2 Z2    ; move 7) Disable ICRC, and lead-off while doing a move
  
```

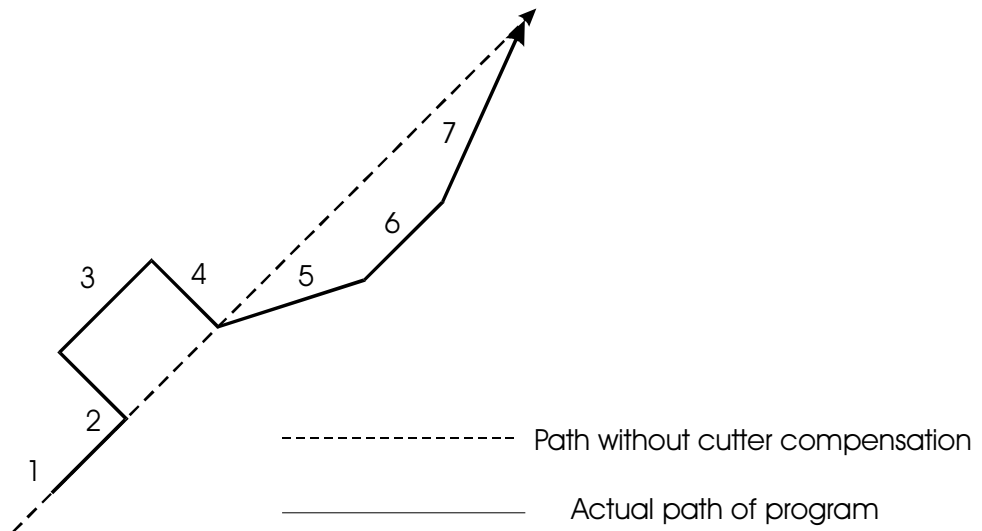


Figure 2-21. Effect of Cutter Compensation on Tool Path

2.8.4. Set Cutter Compensation Radius

G43

By default, the diameter of the cutting tool is assumed to be the value specified in the CNC Initialization screen using the "Tool Diameter" entry field (refer to the *UNIDEX 631/U600 User's Manual*). The G43 command overrides that default.



This command requires one parameter, the tool radius. The unit of measure associated with this radius is either inches or millimeters, dependent upon the current status of the units G-code group (G70/G71).

This command may only be used when cutter compensation is not active.

SYNTAX: *G43 ToolRadius*

EXAMPLE:

G70 G43 R0.1	;Sets the new tool radius to 1/10 of an inch
G71 G43 R0.1	;Sets the new tool radius to 1 millimeter

A G44 must be executed before a G43.



2.8.5. Set Cutter Compensation Axes

G44

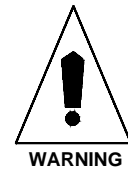
By default, the axis pair to be used for cutter compensation is determined by evaluating the options selected from the Cutter1 and Cutter2 Menus on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*). The G44 command overrides those defaults.



This command defines a new axis pair to be used for cutter compensation, so long as the user specify two axis names and none of the following are true.

SYNTAX: *G44 X Y* or *G44 Z X* or *G44 XZ*

The order of the axis specified is very important. it must match the order specified by the circular plane specification. The first axis in the G44 command must also be the first axis specified in the G17, G18 or G19 command. The same thing applies to the second axis.



EXAMPLE:

G44 X Y	;Designates that axis_01 and axis_02 (X/Y) are to be used for ICRC
G44	;Designates that default axes specified in the CNC configuration screen are used for ICRC

G44 by itself causes the cutter axes from the setup menu to be activated.



2.8.6. Disable Polar or Cylindrical Coordinate Transformation G45

The G45 command disables the polar or cylindrical coordinate transformation.

SYNTAX: *G45*

EXAMPLE:

See example under the G46 command

2.8.7. Enable Polar Coordinate Transformation G46

The G46 command enables a transformation from X/Y cartesian axis plane into a polar coordinate system. The polar coordinate system is comprised of a linear positioning device holding the tool and a rotary device that holds the part centered about its axis of rotation. Parts programming is done using the G1/G2/G3 commands acting upon virtual (no D/A or feedback device defined) X and Y axis. The X/Y axis to be transformed is defined with G44 command.



Cutter compensation may be used in the G46 mode.

SYNTAX: *G46 linear_axis rotary_axis*

The example on the following page is an example of a 4 inch/mm square part with rounded corners (1.0 inch/mm radius) centered at the X/Y origin. Contact with the part will occur at X=2.0, Y=0.0. The rotary axis in the polar coordinate system is C and the linear axis is RAD.

EXAMPLE:

G90 F100 E10.	;Absolute mode
G9 G1 X2.0 Y0.0	;Move X and Y virtual axis to edge of part
G9 G1 RAD ??? C ???	;Align tool to part. Positions are machine dependent
G44 X Y	;Specify the X/Y plane
G46 RAD C	;Enable polar coordinates on RAD and C axis
G91 G1 Y1.0	
G3 X-1.0 Y1.0 I-1.0 J0	
G1 X-2.0	
G3 X-1.0 Y-1.0 IO J-1.0	
G1 Y-2.0	
G3 X1.0 Y-1.0 I1.0 J0	
G1 X2.0	
G3 X1.0 Y1.0 IO J1.0	
G9 G1 Y1.0	;Back at starting point
G9 G1 X 1.0	;Move off of the part
G45	;Disable polar coordinate transformation

The part must always cover the center of rotation of the rotary axis. Motion through the virtual X/Y axis origin with the G46 mode active should not be attempted.

Only G1/G2/G3 commands are valid under polar coordinate transformation. G0 commands or any other axis motion command should not be attempted when G46 is active.



2.8.8. Enable Cylindrical Coordinate Transformation G47

The G47 command enables a transformation from the X/Y cartesian axis plane into a cylindrical coordinate system. The cylindrical coordinate system is comprised of a rotary axis holding the part to be machined and a linear (the X axis in the cartesian coordinate system) that moves parallel to the center of rotation of the rotary axis. An additional axis perpendicular to the center line of rotation of the rotary axis may be present for positioning a tool in the proximity of the part to be machined (this axis is not controlled by the coordinate transformation). Part programming is done using G1/G2/G3 commands acting upon a virtual Y axis (no D/A or feedback device defined) and the X axis. The X/Y axis plane is defined by the G44 command and the CNC parameter page (see Plane Selection under the CNC Parameter item on the Setup Menu). The arguments of the G47 command define the rotational axis and the radius of the part being processed. All subsequent Y axis position commands will refer to circumferential distances around a part of the specified radius according to the following relationship:

$$rotation_position = Y_axis_commanded_position * 360 / (2 * PI * current_radius)$$

Only G1/G2/G3 commands and cutter compensation generated motion are valid under this mode of operation. G0 commands or any other type of axis motion should not be attempted when the cylindrical coordinate transformation is active.



Figure 2-22 is an illustration of the relationship between the X,Y, rotational and optional infeed axis.

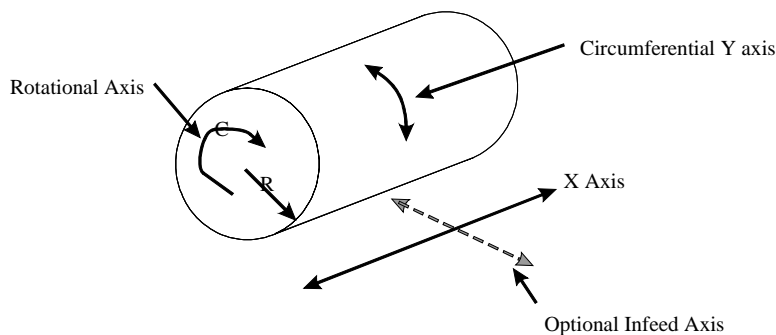


Figure 2-22. X, Y, Rotational and Optional Infeed Axis

SYNTAX: *G47*
 G47 Rotary_Axis R[Current_Radius]

The example program below illustrates the commands required to enable and disable cylindrical coordinate transformation. The axis designations used in this example are consistent with Figure 2-22.

EXAMPLE:

G44 X Y	;set X/Y plane for G47
G47 C R2.0	;enable cylindrical coordinates
G9 G1 X2.0 Y2.0	;perform axis motion
.	;
.	;G1/G2/G3 motion commands
.	;
G45	disable cylindrical coordinates

2.9. Fixture Offset (G53, G54, G55)

The fixture offset feature provides you the ability to program part dimensions relative to a fixed point in space, not knowing the absolute coordinates of that point. Dimensional distances specified in the absolute (G90) mode are relative to this point in space, as opposed to the last software home position.

The UNIDEX 631/U600 CNC provides support for two such points in space. These points are referred to as fixture offset #1 and fixture offset #2. Separate G-codes have been implemented to permit you to determine which fixture offset is currently active.

2.9.1. Cancel Fixture Offset **G53**

The G53 command cancels any fixture offsets currently active. The preset position register(s) of the axes to which the fixture offset is applied are immediately updated to reflect the change in the coordinate system. The machine position registers are unaffected. Please note that only one offset may be active per axis. The offsets are not additive.

SYNTAX: G53

EXAMPLE:

```
G53                                 ;De-activate any fixture offset present in the system
```

The first move after the fixture offset is deactivated must be performed in the absolute (G90) mode. Otherwise, the program position and the preset position registers will not agree. Refer to the comprehensive example following the G55 command description.



2.9.2. Set Fixture Offset #1 **G54**

The G54 command specifies the absolute coordinates of the point referred to as fixture offset #1. The preset position register(s) of the specified axes are updated immediately to reflect the change in the coordinate system. The machine position registers are unaffected.

Refer to the comprehensive example following the G55 command.

SYNTAX: G54 AxisName/Coordinate [AxisName/Coordinate]

EXAMPLE:

```
G54 X10. Y5. Z3.                 ;Enable fixture offset #1. All dimensional data will now be  
                                   ;relative to the point (10,5,3) instead of (0,0,0).
```

The fixture offsets have no effect when operating in the incremental programming (G91) mode.

If a fixture offset is currently in the system when this command is processed, the old fixture offset is removed prior to the activation of the new offset.





2.9.3. Set Fixture Offset #2

G55

The G55 command specifies the absolute coordinates of the point referred to as fixture offset #2. The preset position register(s) of the specified axes are updated immediately to reflect the change in the coordinate system. The machine position registers are unaffected.

Refer to the comprehensive example following this command.

SYNTAX: *G55 AxisName/Coordinate [AxisName/Coordinate]*

EXAMPLE:

G55 X10. Y5. Z3. ;Enable fixture offset #2. All dimensional data will now
;be relative to the point (10,5,3) ;instead of (0,0,0).



The fixture offsets have no effect when operating in the incremental programming mode (G91).

If a fixture offset is currently in the system when this command is processed, the old fixture offset is removed prior to the activation of the new offset.

Fixture Offset Example

Line #	Program Line	Comments	X,Y Preset	X,Y Machine
N10	(Home,X,Y)	Move X and Y axes to hardware home	0,0	0,0
N20	G90 F100.	Use absolute distance mode and set feedrate.	0.0	0.0
N30	G1 X10. Y10.	Move the X and Y axes 10.	10,10	10,10
N40	G92	Set software home	0,0	10,10
N50	G54 X5. Y3.	Activate fixture offset #1 at 5,3	-5,-3	10,10
N60	G1 X10. Y15.	Move the X axis 10.0 and Y axis 15.0	10,15	25,28
N70	G53	De-activate fixture offsets	15,18	25,28
N80	G55 X-10. Y5.	Activate fixture offset #2 at -10,5	25,13	25,28
N90	G1 X10. Y15.	Move to absolute position 10,15	10,15	10,30
N100	G53	De-activate fixture offsets	0,20	10,30
N110	G54 X5. Y5.	Re-activate fixture offset #1 at 5,5	-5,15	10,30
N120	G1 X5. Y5.	Move to absolute position 5,5	5,5	20,20
N130	G55 X10. Y10.	De-activate fixture offset #1 and activate fixture offset #2	0,0	20,20
N140	G1 X-10. Y-10.	Move to absolute position -10,-10	-10,-10	10,10
N150	G53	De-activate all fixture offsets	0,0	10,10

When the fixture offsets were activated (N50, N80 and N110), the value of the fixture offset was subtracted from the value of the preset registers for those axes.

When the fixture offsets were deactivated (N70, N100 and N150), the offset currently being used was added into the value of the preset registers for those axes.

When changing from fixture offset #1 being active to fixture offset #2 being active, the values for fixture offset #1 were added into the preset registers before the values for fixture offset #2 were added into those registers.



2.10. Parameter Monitoring (G56, G57)

The UNIDEX 631/U600 CNC provides a feature by which you may cause axis motion to be terminated upon the detection of a specified condition. This condition must involve one of the axis parameters.

Any axis parameter designated as readable may be used by this command. See appendix A for a list of valid axis parameters.

2.10.1. Enable Parameter Monitoring

G56

The G56 command enables the parameter monitoring feature. The parameters for this command include an axis specification and a conditional expression involving one axis parameter and one numeric literal. The axis specification should be made using the naming convention currently active (e.g., hardnames/softnames). This list may contain any number of axes, but all axes mentioned must be associated with this CNC.

SYNTAX: *G56 AxisList AxisParam RelationalOperator NumericLiteral*

EXAMPLE:

G56 X Y IAVG GT 5000	;Terminate X/Y motion if the average current exceeds 5000
G56 Z POSERR GT 100	;Terminate Z axis motion if the position error exceeds 100

Only one parameter monitoring function may be active simultaneously.



2.10.2. Disable Parameter Monitoring

G57

As mentioned previously, the UNIDEX 631/U600 CNC provides a feature by which you may cause axis motion to be terminated upon the detection of a specified condition. This mode of operation is referred to as parameter monitoring.

The G57 command cancels all parameter monitoring being performed by the system. Therefore, it cancels the effect of any G56 commands which were previously active. This command accepts no parameters.

SYNTAX: *G57*

EXAMPLE:

G57	;Terminate all parameter monitoring presently active in the system
-----	--

2.11. Acceleration/Deceleration Overview (G60, G61)

The trajectory generator provides several types of axis acceleration and deceleration. You may choose the type which is most suited to the mechanics of the system.

As can be seen from the CNC Initialization screen, the UNIDEX 631/U600 CNC has several parameters which control the acceleration and deceleration of all motion. The Accel/Decel Control Group Box contains parameters for Accel Time, Decel Time, Accel Rate and Decel Rate. The G-codes Menu provides options of Time-vs-Rate Based and as well as Linear-vs-Sinusoidal profiles.

The time-vs-rate based parameter determines the main operational mode of the trajectory generator. While operating in a rate based mode (G68), the Accel Rate and Decel Rate parameters are used to control the acceleration and deceleration of the axes. These parameters are specified in user units (in, mm or deg) per second per second and therefore result in the generation of a linear acceleration (see Figure 2-23). When operating in a time based mode (G67), the Accel Time and Decel Time parameters specify the amount of time in which the axes are to execute changes in velocity.

The CNC also provides the flexibility of choosing between two different types of acceleration and deceleration while operating in this mode: linear (G63) and sinusoidal (G64) (1-cosine). Figure 2-23 demonstrates the effect of each upon the velocity curve of a single axis.

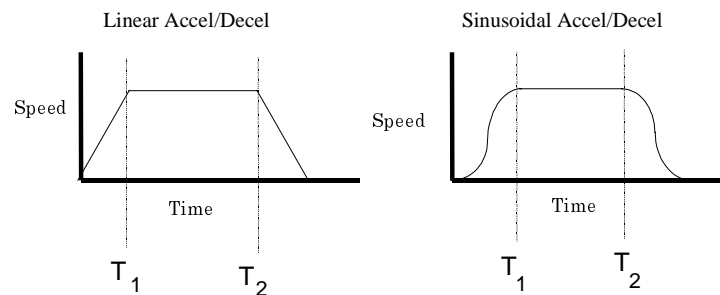


Figure 2-23. Constant Acceleration vs. 1-Cosine



If a G-code completes the move to the specified distance before its acceleration is completed, then the acceleration within that G-code will be linear regardless of the G60/G61 setting.

Sinusoidal acceleration is typically used on systems containing a large inertial mass which is resistant to sudden changes in acceleration. As illustrated, the motion accelerates gradually, then accelerates steeply. As it approaches the commanded velocity, the acceleration gradually decreases until it reaches zero. This reduces jerk associated with discontinuous acceleration such as in the linear Accel/Decel example.

Regardless of whether sinusoidal or linear mode is chosen, the acceleration and deceleration will be completed in the same amount of time.



There are a number of conditions where the controller cannot obey the specified acceleration and may generate an instantaneous deceleration of the velocity command. See G8 and G9 for details.



2.11.1. Set Acceleration Time

G60

When performing acceleration and deceleration using the time based (G67) parameters, the UNIDEX 631/U600 CNC uses the Accel Time parameter specified within the Accel/Decel Control Group Box found on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual* for more information).

The G60 command overrides the default acceleration time parameter for this CNC. Subsequent motion commands accelerate to the commanded velocity within the specified time period. This time period is specified in seconds, with a resolution of 0.001 seconds (1 millisecond).

SYNTAX: *G60 FAccelTime*

EXAMPLE:

G60 F0.5	;Sets new acceleration time to 1/2 sec (500 msec)
----------	---

The parameter may be set regardless of the current setting of the Ramp Type G-code group. However, the effect of this parameter will be apparent only when operating in a time based (G67) acceleration mode.



2.11.2. Set Deceleration Time

G61

When accelerating or decelerating using the time based (G67) parameters, the UNIDEX 631/U600 CNC uses the Decel Time parameter specified within the Accel/Decel Control Group Box found on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual* for more information).

The G61 command overrides the default deceleration time parameter for this CNC. Subsequent motion commands decelerate to the zero velocity within the specified time period. This time period is specified in seconds, with a resolution of 0.001 seconds (1 millisecond).

SYNTAX: *G61 FDecelTime*

EXAMPLE:

G61 F0.25	;Sets new deceleration time to 1/4 sec (250 msec)
-----------	---



The parameter may be set regardless of the current setting of the Ramp Type G-code group. However, the effect of this parameter will be apparent only when operating in the time based (G67) acceleration mode.

2.11.3. Set Profile Time

G62

When processing any motion command, the UNIDEX 631/U600 CNC calculates the desired position of each axis at equally spaced intervals in time. It then applies a splining algorithm on those points to provide a smooth transition between points.

The G62 command changes the time interval at which these points are calculated. By default, this interval is ten milliseconds (10 msec). This default is sufficient for the vast majority of users. However, you have the capability of changing this if necessary.

For example, if you have all four CNC's executing parts programs simultaneously and each program is composed of many motion blocks which are short in duration, the axis processor card may run out of processor band-width. In which case, the G62 command would be used to increase the amount of time between points, thereby decreasing the number of points and the processing time required for each program block.

An application which requires high accuracy velocity profiling might decrease this time interval, thereby increasing the number of points used to generate the desired path.

The only parameter for this command is specified using the F keyword and is the new time interval in seconds. The resolution of this parameter is 0.001 and must be greater than zero.

SYNTAX: *G62 Fxxx* (where xxx is the time interval in seconds)

EXAMPLE:

G62 F0.005	;Sets the new profile time to 5 milliseconds
------------	--



2.11.4. Sinusoidal (1-Cosine) Acceleration Mode

G63

When accelerating and decelerating using the time based parameters (G67), the UNIDEX 631/U600 CNC offers the flexibility of choosing between two distinct types of acceleration; linear or sinusoidal. The G63 command specifies that the acceleration type to be used is sinusoidal.

Sinusoidal acceleration is typically used on systems containing a large inertial mass which is resistant to sudden changes in acceleration.

Please refer to the Acceleration/Deceleration Overview.

SYNTAX: *G63*



This is the default operational mode of acceleration while operating from time based parameters.

The parameter may be set regardless of the current setting of the Ramp Type (G67/G68) G-code group. However, the effect of this parameter will be apparent only when operating in time based (G67) acceleration mode.

2.11.5. Linear Acceleration Mode

G64



When accelerating and decelerating using the time based parameters (G67), the UNIDEX 631/U600 CNC offers the flexibility of choosing between two distinct types of acceleration: linear or sinusoidal. The G64 command specifies that the acceleration type to be used is linear.

Linear acceleration is typically used on systems which require constant rates of acceleration within each move.

Please refer to the Acceleration/Deceleration Overview.

SYNTAX: *G64*



The parameter may be set regardless of the current setting of the Ramp Type (G67/G68) G-code group. However, the effect of this parameter will be apparent only when operating in the time based (G67) acceleration mode.

2.11.6. Set Acceleration Rate**G65**

When accelerating and decelerating using the rate based parameters (G68), the UNIDEX 631/U600 CNC uses the Accel Rate parameter specified within the Accel/Decel Control Group Box found on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual* for more information).

The G65 command overrides the default acceleration time parameter. Subsequent motion commands will accelerate to the commanded velocity using the rate specified. This rate is specified in user units/second/second.

If rate based acceleration is chosen, then that rate is applied over the vector distance of the move. For example, if a G70 G90 X1 Y1 is executed, then the specified rate will be correct over the distance of 1.414 inches, not 1 inch.

The user units of acceleration can be in mm, inches or degrees depending on the situation. If all the axes in the move are linear, then the units will be inches or millimeters based on the settings of G70/G71. If all the axes in the move are rotary, the user units are in degrees. If the move contains both rotary and linear axes, then the user units will be those of the dominant axis. Refer to G98 for details on what dominance is to see how these types of moves are accomplished.

SYNTAX: *G65 Fxxxx* (where xxxx is either inches/sec/sec or millimeters/sec/sec)

EXAMPLE:

G70 G65 F0.5	;Sets the new acceleration rate to 0.5 inches/ second/second
G71 G65 F1.27	;Sets the new acceleration rate to 1.27 mm/ second/second

The parameter may be set regardless of the current setting of the Ramp Type G-code group. However, the effect of this parameter will be apparent only when operating in the rate based (G68) acceleration mode.



2.11.7. Set Deceleration Rate**G66**

When accelerating and decelerating using the rate based parameters (G68), the UNIDEX 631/U600 CNC uses the Decel Rate parameter specified within the Accel/Decel Control Group Box found on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual* for more information).

The G65 command overrides the default deceleration time parameter for the UNIDEX 631/U600 CNC. Subsequent motion commands accelerate to the commanded velocity using the rate specified. This rate is specified in either inches/second/second or millimeters/second/second, depending upon the current setting of the units (G70/G71) G-code group.

SYNTAX: *G66 Fxxxx* (where xxxx is either inches/sec/sec or millimeters/sec/sec)

EXAMPLE:

G70 G66 F0.1	;Sets the new deceleration rate to 0.1 inches/second/second
G71 G66 F0.254	;Sets the new deceleration rate to 1.27 mm/second/second



The parameter may be set regardless of the current setting of the Ramp Type G-code group. However, the effect of this parameter will be apparent only when operating in the rate based (G68) acceleration mode.

2.11.8. Acceleration/Deceleration Time Based (Ramp Type) G67

The UNIDEX 631/U600 offers two modes of operation with respect to acceleration and deceleration: time or rate based. The G67 command specifies that acceleration and deceleration are to be performed with time based parameters.

While operating in this mode, the Accel Time and Decel Time parameters are used to specify the amount of time in which all moves are to accelerate to and decelerate from the commanded velocity. These parameters may be changed using the G60 and G61 commands, respectively. The current setting of the Accel Mode (G63/G64) G-code group determines the type of acceleration to be performed (linear or sinusoidal).

Please refer to the Acceleration/Deceleration Overview (Section 2.11.).

SYNTAX: G67

This is the default operational mode of acceleration and deceleration.

The word "MODAL" is written in a bold, stylized font with a 3D effect. The letters are black with a yellow-to-orange gradient and a shadow effect, giving it a metallic or glowing appearance.**2.11.9. Acceleration/Deceleration Rate Based (Ramp Type) G68**

The UNIDEX 631/U600 offers two modes of operation with respect to acceleration and deceleration: time and rate based. The G68 command specifies that acceleration and deceleration are to be performed based upon rate based parameters.

While operating in this mode, the Accel Rate and Decel Rate parameters are used to specify the amount which the velocity is to change each second. Therefore, the amount of time used to reach the commanded velocity varies between moves. These parameters may be changed using the G65 and G66 commands, respectively. The type of trajectory generated is always linear.

Rate based acceleration and deceleration is typically used on systems which are limited in acceleration and are commanded to varying velocities.

Please refer to the Acceleration/Deceleration Overview (Section 2.11.).

SYNTAX: G68

The word "MODAL" is written in a bold, stylized font with a 3D effect. The letters are black with a yellow-to-orange gradient and a shadow effect, giving it a metallic or glowing appearance.



2.12. Programming Codes

2.12.1. Inch Dimension Programming Mode (Units)

G70

The UNIDEX 631/U600 CNC provides you the option of specifying all distances and feedrates in either English units (inches) or metric units (millimeters). The G70 command indicates English units.

While the G70 mode is active, all distances are specified in inches, all speeds are in inches per second and all acceleration rates are specified in inches per second per second.

SYNTAX: G70

EXAMPLE:

G70.	;Set English programming mode (inches)
G1 X10.	;Move the X axis 10 inches in the positive direction
G1 Y-5.	;Move the Y axis 5 inches in the negative direction
F100.	;A feedrate of 100 inches per minute is established



The default mode of operation is established using the G-codes Menu found within the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).

The number of machine counts per inch is specified using the "inches per motor revolution" parameter.

2.12.2. Metric Dimension Programming Mode (Units)

G71

The UNIDEX 631/U600 CNC provides you the option of specifying all distances and feedrates in either English units (inches) or metric units (millimeters). The G71 command indicates that units are metric.

While the G71 mode is active, all distances are specified in millimeters, all speeds are in millimeters per second and all acceleration rates are specified in millimeters per second per second.

SYNTAX: G71

EXAMPLE:

G71	;Set metric programming mode (millimeters)
G1 X4.	;Move the X axis 4 mm in the positive direction
G1 Y-2.	;Move the Y axis 2 mm in the negative direction
F100.	;A feedrate of 100 mm per second is established

The default mode of operation is established using the G-codes Menu in the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).

The number of machine counts per inch is specified using the "millimeters per motor revolution" parameter.



2.12.3. Restore Position Registers

G82

The G82 command restores position registers to the value prior to executing a G92 command.

SYNTAX: G82

EXAMPLE:

G82	;Restore all axes registers
G82 Y, Z	;Restore the Y and Z axes only

Do not use G82 when in the G42 or G43 mode, the results are unpredictable.





2.12.4. Mirror Image

G83

The G83 command activates the mirror image function. The UNIDEX 631/U600 mirror image is available for all axes. The G83 command operates in both the incremental and absolute mode. While in the absolute mode however, the specified positions 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.

Refer to Figure 2-24 for a mirror image example.

SYNTAX: *G83, axis name, axis name*

The mirror function is disabled with the G83 command. This statement must occupy its own line (no other CNC blocks on this line).

EXAMPLE:

G83	;Disables mirroring
G70	;Set English programming (inches).
(REF, X, Y)	;Establish a software home for X and Y axes
G83 X1. Y1.	;Activate mirror image function for X and Y axes, changing positive X and Y values to negative values
G91	
(CLS, BOX1)	;Call subroutine BOX1
G83	;Disables mirroring
M2	;Stop the program
(DFS, BOX1	;Define subroutine BOX1
X2. Y4.	;Initiate a positive linear move for the X and Y axis
F100.	;Establish a feedrate of 100 in. per minute
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. Y-4.	;Initiate linear move of X and Y axis to home position
)	;end of extended command block
M0	;Program stop (see Figure 2-24)

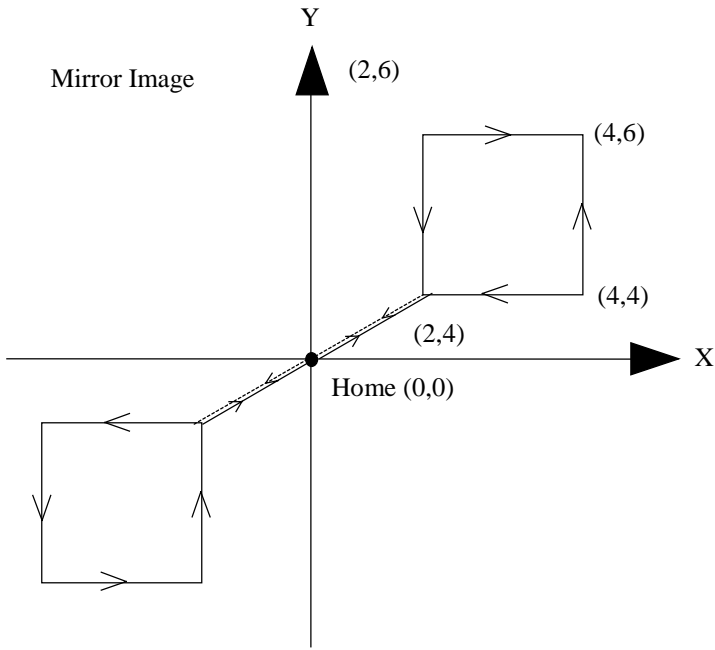


Figure 2-24. G83 Mirror Image Example

2.12.5. Parts Rotation

The G84 command is used to define the plane and angle of part rotation. The parts rotation programming feature permits the changing of orientation (in a plane) of a sequence of moves without changing the move coordinates or changing coordinate reference frames.

Refer to Figure 2-25 for a parts rotation example.

SYNTAX: (*G84, axis1, axis2, plane for rotation, angle*)

The angle required for entry must be converted to degrees and may be a user degree's specification.

The angle specified is relative and is based on the last angle set in a G84 command (see example).

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 specified as zero will deactivate parts rotation. This statement block must occupy its own line (no other blocks on the line).

EXAMPLES:

G84 X, Y, 45	Part rotation angle is set to 45° in XY plane
G84 X, Y, VAR1	Part rotation angle is set to value of VAR1
G84 X, Y, 0	Deactivate parts rotation

LINE	COMMAND	DESCRIPTION
N1	G1 X5. F1000.	;Move X axis five units in plus direction
N2	G84 X, Y, 45	;Set X, Y part rotation angle to 45°
N3	X5.	;Move X axis five units in plus direction
N4	G84 X, Y, -90	;Set the X, Y part rotation angle to -45°
N5	X5.	;Move X axis five units in plus direction
N6	G84 X, Y, -90	;Set the X, Y, part rotation angle to 225°
N7	X5.	;Move X axis five units in plus direction
N8	G84 X, Y, +300	;Set the part rotation angle to 390° (30°)
N9	X5.	;Move X axis five units in plus direction
N10	G84 X, Y, -30	
N11	X5.	;Move X axis five units in plus direction
N12	G84, X, Y, 0	;Disable part rotation
N13	M2	;End of the program (see Figure 2-25)

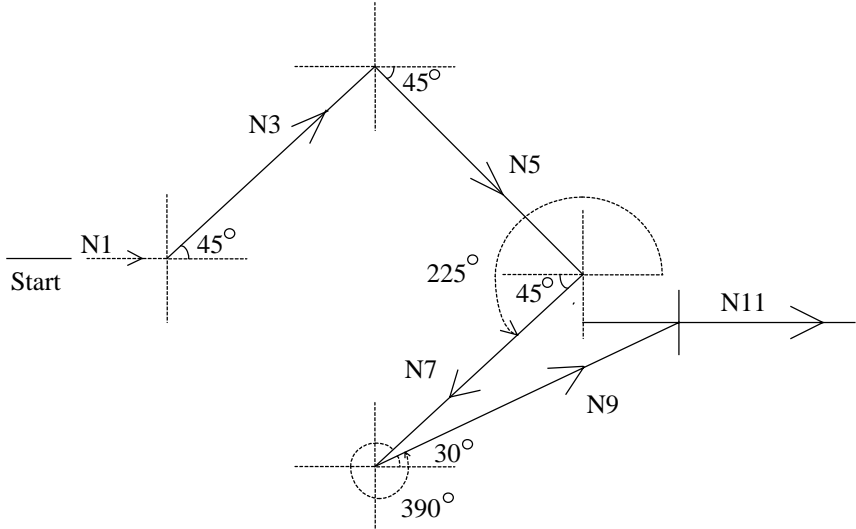


Figure 2-25. G84 Parts Rotation Example



2.12.6. Absolute Dimension Programming Mode (Distance) G90

Prior to the execution of motion commands, the UNIDEX 631/U600 CNC must be told whether programmed dimensional data is to be interpreted as absolute coordinates, or as an offset from the current axis position. The G90 command specifies that all positions should be interpreted as absolute coordinates.

SYNTAX: G90

EXAMPLE:

Assume the axes start at (0,0). Figure 2-26 illustrates the results of this example.

G90	;Set absolute programming mode
F100.	;Establish feedrate for subsequent moves
G1 X10.0 Y10.0	;Move X and Y axes to absolute coordinate 10,10
G1 X15.0 Y25.0	;Move X and Y axes to absolute coordinate 15,25
G1 X15.0 Y10.0	;Move X and Y axes to absolute coordinate 15,10

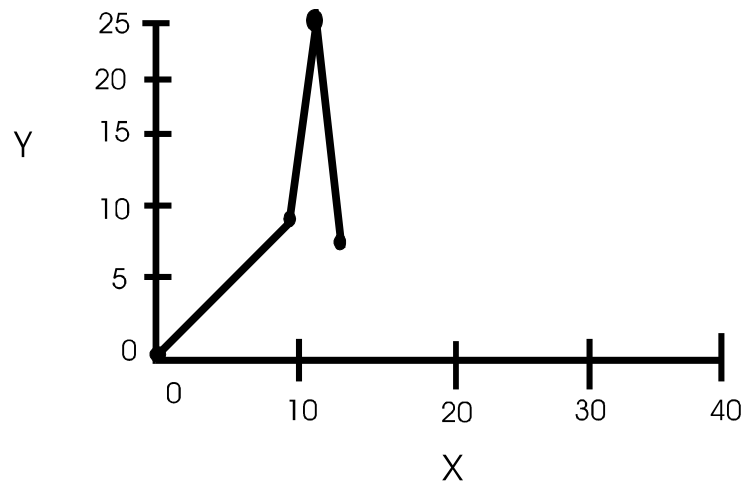


Figure 2-26. Absolute Mode Programming



The default mode of operation is established using the G-codes Menu in the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).

2.12.7. Incremental Position Programming (Distance)

G91

Prior to the execution of motion commands, the UNIDEX 631/U600 CNC must be told whether programmed dimensional data is to be interpreted as absolute coordinates or as an offset from the current axis position. The G91 command specifies that all positions should be interpreted as incremental distances from the current position.



SYNTAX: G91

EXAMPLE:

Assume that the starting position is (0,0). Figure 2-24 illustrates the result of this example.

G91	;Set incremental programming mode
F100.	;Establish feedrate for subsequent moves
G1 X10.0 Y10.0	;Move the X and Y axes a distance of 10.0
G1 X15.0 Y25.0	;Move X axis 15.0 and Y axis 25.0 from current position
G1 X15.0 Y10.0	;Move the X axis 15.0 and Y axis 10.0 from current position

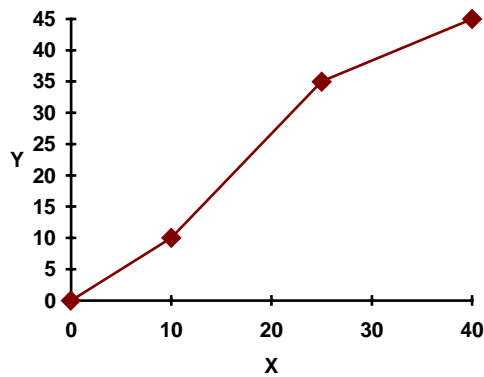


Figure 2-27. Incremental Mode Programming

The default mode of operation is established using the G-codes Menu in the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).





2.12.8. Software Home (Set Position Registers)

G92

This command changes the value of the preset position registers. These preset registers determine the current position when executing motion commands in the absolute dimension (G90) programming mode.

As can be seen from the syntax description below, axes' positions may also be specified within this program block. If specific axes' positions are present, the preset registers for those axes specified are set to the value associated within the program block. In the event that an axis name is mentioned, without a corresponding position, the preset register of that axis will be set to zero. Axes which are not specified will not be affected.

The user must provide a set of axes as parameters to the G92 command.

SYNTAX: *G92 [AxisName AxisPos] [...] [AxisName AxisPos]*

EXAMPLE:

G92 X1.0 Y2.0	;Sets the preset register of the X axis to 1.0 and the Y axis to 2.0
G92 X10. Y0.	;Sets the preset register of the X axis to 10.0 and the preset register of the Y axis to zero



The G92 command changes the value of the preset position registers, but not the value of the machine position registers. The machine position registers always reflect the distance from the hardware home position.

The G82 command clears these presets.



Do not use G92 when in the G42 and G43 mode, the results are unpredictable.

2.13. Feedrate and Spindle Speed Codes

2.13.1. Inverse Time Feedrate Programming (FeedrateMode) G93



The UNIDEX 631/U600 CNC provides the flexibility to program feedrates in either units/minute or minutes/unit. The G93 command specifies that feedrates should now be interpreted as minutes per unit. The feedrate calculation is shown below.

$$F = \frac{\text{\# of minutes to complete move}}{\text{SquareRoot}(X^2 + Y^2 + \dots + a^2)}$$

where: X is the move distance for the X (linear) axis
 Y is the move distance for the Y (linear) axis.

When performing circular interpolation on two axes, the sum of the squares for those axes is replaced by the product of the arc radius and the arc angle squared. For example, if circular interpolation is being performed on the X and Y axes, the feedrate would be set as follows:

$$F = \frac{\text{\# of minutes to complete move}}{\text{SquareRoot}((R * \text{Theta})^2 + Z^2 + \dots + a^2)}$$

where: R is the arc radius
 Theta is the arc angle (in radians)
 Z is the move distance for the Z axis
 a is the move distance for the a axis.

SYNTAX: G93

EXAMPLE:

G93	;Set inverse time feedrate programming
G70 F1.0	;Establish feedrate of 1 minute per inch
G71 F0.5	;Establish feedrate of 0.5 minutes per millimeter

This is not the default operational mode of the CNC.



G93 has no effect on the speed of the rotary axes (E feedrate). The E feedrate is always in units of RPM.





2.13.2. Feed Per Minute Feedrate Programming (FeedrateMode) G94

The UNIDEX 631/U600 CNC provides the flexibility to program feedrates in either units/minute or minutes/unit. The G94 command specifies that feedrates should be interpreted as units per minute. The feedrate calculation is shown below.

$$F = \frac{\text{SquareRoot}(X^2 + Y^2 + \dots + a^2)}{\# \text{ of minutes to complete move}}$$

where: X is the move distance for the X (linear) axis
Y is the move distance for the Y (linear) axis, etc.

When performing circular interpolation on two axes, the sum of the squares for those axes is replaced by the product of the arc radius and the arc angle squared. For example, if circular interpolation is being performed on the X and Y axes, the feedrate would be set as follows:

$$F = \frac{\text{SquareRoot}((R * \text{Theta})^2 + Z^2 + \dots + a^2)}{\# \text{ of minutes to complete move}}$$

where: R is the arc radius
Theta is the arc angle (in radians)
Z is the move distance for the Z axis
a is the move distance for the a axis.

SYNTAX: G94

EXAMPLE:

G94	;Set normal feedrate mode
G70 F100.	;Establish feedrate of 100.0 inches per minute
G71 F500.	;Establish feedrate of 500 millimeters per minute



This is the default operational mode of the controller.



G94 has no effect on the speed of the rotary axes (E feedrate). The E feedrate is always in units of RPM.

2.13.3. Feed Per Spindle Revolution Feedrate Programming G95



The UNIDEX 631/U600 CNC permits the actual velocity of the spindle axis to control the velocity command of a linear axis. If the spindle speed varies during the machining process, the speed of the associated linear axis will vary also.

The G95 command specifies that the speed of the spindle is to be used to determine the desired velocity of the other axes in motion. The feedrate keyword (F) contains the vectorial distance which the slave axes are to be moved for each revolution of the spindle axis. This feedrate can be calculated as follows:

$$F = \frac{\text{SquareRoot}(X^2 + Y^2 + \dots + a^2)}{\# \text{ of spindle revs to complete move}}$$

where: X is the move distance for the X (linear) axis
 Y is the move distance for the Y (linear) axis.

When performing circular interpolation on two axes, the sum of the squares for those axes is replaced by the product of the arc radius and the arc angle squared. For example, if circular interpolation is being performed on the X and Y axes, the feedrate would be set as follows:

$$F = \frac{\text{SquareRoot}(R * \text{Theta}^2 + Z^2 + \dots + a^2)}{\# \text{ of spindle revs to complete move}}$$

where: R is the arc radius
 Theta is the arc angle (in radians)
 Z is the move distance for the Z axis
 a is the move distance for the a axis.

The portion of this vectorial distance attributed to each axis is proportional to the percentage of the total vectorial move distance attributed to this axis.

Actual feedrate in user units is: $\left[\text{ipm} \sqrt{\text{mmpm}} \right] F \left[\frac{\text{user units}}{\text{rev}} \right] * S \left[\frac{\text{rev}}{\text{min}} \right]$

G95 has no effect on the speed of the rotary axes (E feedrate). The E feedrate is always in units of RPM.



Whne in this mode, the programmed feedrate (F word) has no effect, since the linear feedrate is controlled by the spindle.



SYNTAX: G95

EXAMPLE:

G95	;Set feed per spindle revolution feedrate ;programming
G70 G1 X10. F1.0	;The linear axis X is to be moved 10 inches, ;at a speed of one inch per spindle ;revolution. Therefore this move will be ;completed in 10 spindle revolutions.
G71 G1 Y6.0 F0.5	;The linear axis Y is to be moved 6 ;millimeters, at a speed of 1/2 millimeter per ;spindle revolution. Therefore, this move will ;be completed in 12 spindle revolutions.
G70 G1 X2.0 Y4.0 F2.235	;The linear axes X and Y are commanded to ;move 2 and 4 inches, respectively. For each ;revolution of the spindle, the vectorial ;distance to move the slaves is 2.235 inches. ;Therefore, the move will be completed in two ;spindle revolutions. ;Since X is traveling 1/2 the distance of Y, X ;will move 0.745 inches per spindle ;revolution and Y will travel 1.490 inches.



This is not the default operational mode of the controller.

2.13.4. Constant Surface Speed Spindle Programming**G96**

In some cutting applications, it is desirable to allow the speed of the spindle to be driven by the position of an axis perpendicular to the spindle.



The G96 command enables this mode of operation. It is referred to as "Constant Surface Speed" spindle programming because as the perpendicular axis comes closer to the center of the part, the speed of the spindle is increased. As it moves further away, the spindle speed is decreased. The amount of spindle speed differential is proportional to the change in surface area of the part being cut. That is, as the radius of the part decreases, the speed of the spindle is increased to maintain a constant surface velocity. The inverse of this is also true.

Consider an application removing material from the top of a cylindrical object. In order to remove the maximum amount of material in a given time period, the spindle needs to move as quickly as the tool can remove the material. Since the amount of material to be removed in one spindle revolution is much greater at the outside edge of the cylinder than at the center, the speed of the spindle must be much slower when the tool is located near the outside edge.

The implementation of this feature in the UNIDEX 631/U600 CNC requires that you specify the current distance of the tool tip from the center point of the spindle and the desired surface speed, in units (in or mm) per minute. This information must be specified in the form of parameters to this command. The R keyword specifies the current distance from the tool tip to the center point of the spindle (nominal radius). The SF keyword specifies the desired surface speed, in user units (in or mm) per minute.

It is important to specify the axes used to determine the tool tip coordinate. These must be specified as "threadx" and "thready" axis in CNC init parameters.

The spindle will not exceed the maximum and minimum SF rate as specified in the CNC initialization screen. No warning is delivered if the feedrate is clamped at the minimum or maximum.



SYNTAX: *G96 Rxxxx SFxxxx*
 (where xxxx with R specifies the current distance and xxxx
 with SF specifies the desired surface speed)

EXAMPLE:

G96 R1.0 SF40.	;Places the spindle axis into constant surface ;speed mode. The tool is currently located 1.0 ;unit (in/mm) from the center point of the ;spindle. The spindle speed will be ;maintained such that 40.0 units (in/mm) of ;the part will pass below the tool each minute.
M03	;Turn on the spindle



This is not the default operational mode of the CNC.

An axis must be designated as the spindle axis via the Spindle Menu on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).

The axis which is perpendicular to the spindle must be defined as the axis chosen as the "ThreadY" axis on the CNC Initialization screen.

2.13.5. Direct RPM Spindle Programming**G97**

The G97 command specifies that the spindle is to be controlled by you via the S keyword. While operating in this mode, the units associated with this keyword are revolutions per minute. Once turned on (M03), the spindle speed responds immediately to changes in the value of this keyword.

SYNTAX: G97

EXAMPLE:

S100.	;Command the spindle to rotate at 100 revolutions per minute
-------	--



This is the default operational mode of the CNC.

An axis must be designated as the spindle axis via the Spindle Menu on the CNC Initialization screen (refer to the *UNIDEX 631/U600 User's Manual*).

The axis which is perpendicular to the spindle is defined as the axis chosen as the "ThreadY" axis on the CNC Initialization screen.

MODAL

2.14. Dominant Feedrate Overview (G98, G99)

A complex move is one in which both linear and rotary axes are moving simultaneously. These types of moves create a number of special problems and the programmer must read and understand the following in order to properly perform complex moves. The first problem is that both the F and the E word speeds cannot be obeyed at once. Refer to the following example code fragment:

```
G90 G0 X0 B0      ; Goto {0,0}; use absolute coordinates from now on
G1 X10 B90 F10 E1
```

If the F word is obeyed, then the G1 move finishes in 1 minute. However, if the E word is obeyed, the move finishes in 15 seconds. The solution the Aerotech controller uses for this problem is to have a mode that specifies which word will be obeyed. In the G99 mode (linear dominant) the E word is ignored during moves involving both rotary and linear axes. Conversely, in the G98 (rotary dominant) mode, the F word speed is ignored in complex moves.

A related item is the acceleration during complex moves. The dominant type of movement (rotary or linear) will use the acceleration as instructed by the codes G63 through G68. However, the non-dominant move will be a slave to the dominant movement and may not obey the acceleration parameters. If the acceleration ramping is time based (see G67), then both types will obey the acceleration time. However, if it is rate based (G68) then the non-dominant axis will not follow any specified acceleration.

Keep in mind that regardless of the G99/G98 mode, if a CNC block moves only linear or rotary axes, then the F and the E words, respectively, will be obeyed. G98/G99 applies only to complex CNC blocks; ones that move rotary and linear axes at once.

See the example below:

(in all examples here, X,Y,Z are linear axis, while B are rotary axis)

Suppose for all three examples we begin at X=0, B=0, in G90 mode.

```
G99 G90 X1 B1    ; Here we use the F feedrate, because we are in G99 mode.
G99 G90 B1      ; Here we use the E feedrate, even though we are in G99
                ;mode.
G99 G90 X0 B1   ; Here we use the E feedrate, even though we are in G99
                ;mode.
```

The second and third examples are equivalent, even though a X is specified in the third example, because the X target is the same as our current location, no X motion will be made. Unfortunately there is an additional problem when using complex moves, due to the fact that the feedrate of the non-dominant axes is totally dictated by the dominant axis. The speed of the non-dominant axis may be exceeded.

Examine the following code fragment, where X is a linear axis, B a rotary axis.

```
G90 G0 X0 B0      ; Goto {0,0}; use absolute coordinates from now on
G99              ; Set Linear dominance
G1 X0.00016666 B180 E1 F1    ; (0.00016666 = one over 6000)
```

Here the user specified linear dominance in the G1 command line, so the linear part of the move must travel at the linear feedrate. However, since the distance the X axis travels is small (relative to the B axis distance) the move must complete in a relatively short time (1/100 second in this case). This forces the B axis to travel at 3000 RPM, which may be

too fast. As discussed in the limiting of feedrates (see F word section in Chapter 1), the U31 controller in this case will scale down the velocities of the axes so that the E word is not exceeded. In the example above, the X axis would travel at 0.02 units/minute, the B axis at 1 RPM, and the move would take 1/2 second.

However, there is a related case where the controller cannot limit the speed properly and it throws a CNC fault. For example, look at the following code fragment, where X is a linear axis, B a rotary axis.

```
G90 G0 X0 B0           ; Goto {0,0}; use absolute coordinates from now on
G99 G1 X10 E1 F100     ; X travels at 100 units/min.
G1 X20 B90 E1 F100     ; X travels at 40 units/min., B at 1 RPM
```

In the first move X travels at 100 units/min and in the second move, the E feedrate limits the speed, forcing the X to slow down to 40 units/min. Since no G9 is specified, the moves must be blended and the X will begin to slow down to 40 units/min. as soon as the second move starts. However, at the moment the second move begins, the X axis will be traveling at 100 units/min. So in order for the B axis to finish the move at the same time, the B axis must travel at 2.5 RPM at the instant the move starts. This would violate the E word limit of 1 RPM, so the controller throws a CNC fault: "Non dominant feedrate exceeded in contoured move". This situation can only be avoided by reducing the F feedrate in both moves or by placing a G9 on the first move, so the X axis decelerates to a stop before beginning the second move.

2.14.1. Rotary Feedrate Dominant

G98

The G98 command specifies that the rotary feedrate (E) is to be considered dominant in coordinated motion commands moving both linear and rotary axes. When operating in this mode, the value of the E keyword determines the move duration and the corresponding feedrate for the linear axis is computed from the duration.

Please refer to the Dominant Feedrate Overview (Section 2.14.) for more information.

SYNTAX: G98

EXAMPLE:

G98, G91	;Make E-feedrate dominant
G1 X10. Y72. F100. E10.	;Assuming X is linear and Y is rotary use E-feedrate to ;calculate move time of 0.2 minutes. Linear feedrate used will ;be 50 units/minute.

An axis is designated as linear from within the Machine Parameter Set-up screen using the "Linear or Rotary" parameter.



2.14.2. Linear Feedrate Dominant**G99**

The G99 command specifies that the linear feedrate (F) is to be considered dominant in coordinated motion commands involving both linear and rotary axes. When operating in this mode, the value of the F keyword determines the move duration and the corresponding feedrate for the rotary axis is computed.

Please refer to the Dominant Feedrate Overview (Section 2.14.) for more information.

SYNTAX: *G99*

EXAMPLE:

G99	;Make F-feedrate dominant.
G1 X10. Y72. F100. E10.	;Assuming X is linear and Y is rotary, use F-feedrate to calculate ;a move time of 0.10 minutes. Rotary feedrate used will be 2 RPM.



An axis is designated as linear from within the Machine Parameter Set-up screen using the "Linear or Rotary" parameter.

2.15. Circular Direction Codes

2.15.1. Normal Circular Interpolation

G110

The UNIDEX 631/U600 CNC provides you with the flexibility to change the orientation of the axes being used for circular interpolation. By default, circular interpolation occurs as described in the description of the G2 and G3 commands, Plane Select (G27/G28/G29) G-code groups and the I/J/K keywords. Table 2-4 defines these relationships.



Table 2-4. G-Codes to Change Axes Used for Circular Interpolation

G-code	Description
G2/G12	Clockwise Arc Direction
G3/G13	CounterClockwise Arc Direction
G17/G27	Plane_X_Axis and Plane_Y_Axis
G18/G28	Plane_Z_Axis and Plane_X_Axis
G19/G29	Plane_Y_Axis and Plane_Z_Axis
I	Plane_X_Axis
J	Plane_Y_Axis
K	Plane_Z_Axis

The G110 command resets these associations if a G111 command had been previously executed.

SYNTAX: *G110*

EXAMPLE:

G110	;Reset normal circular interpolation mode
------	---

This is the default operational mode of the CNC.





2.15.2. Inverse Circular Interpolation

G111

The UNIDEX 631/U600 CNC provides the flexibility to change the orientation of the axes being used for circular interpolation. By default, circular interpolation occurs as described in the description of the G2 and G3 commands, the Plane Select (G27/G28/G29) G-code groups and the I/J/K keywords.

However, when operating in the inverse circular interpolation mode, the arc direction, plane and circle centerpoint keyword associations are reversed. Table 2-5 summarizes the relationships that exist.

Table 2-5. Relationship of Arc Direction, Plane, & Circle Centerpoint

G-code	Description
G2/G12	Counter Clockwise Arc Direction
G3/G13	Clockwise Arc Direction
G17/G27	Plane_Y_Axis and Plane_X_Axis I associated with Plane_Y_Axis J associated with Plane_X_Axis
G18/G28	Plane_X_Axis and Plane_Z_Axis K associated with Plane_X_Axis I associated with Plane_Z_Axis
G19/G29	Plane_Y_Axis and Plane_Z_Axis J associated with Plane_Z_Axis K associated with Plane_Y_Axis

The G111 command sets these associations.

SYNTAX: *G111*

EXAMPLE:

G111	;Set inverse circular interpolation mode
------	--



This is not the default operational mode of the CNC.

2.15.3. 4 Kilo-Hertz Servo Update Rate**G130**

The G130 command causes the U600 to update the servo loop 4000 times/sec (every .25 msec). This includes sampling the encoder feedback and calculating a new I command for the motor. When switching from 1K to 4K update, (G131 to G130) the user must adjust the gains as shown below.

$$\begin{aligned}K_p &= K_p * 4 \\PGAIN &= PGAIN / 4 \\K_i &= K_i\end{aligned}$$

The default mode of operation on power-up may be set in the CNC Initialization screen for each CNC under the G-codes pull down menu.

2.15.4. 1 Kilo-Hertz Servo Update Rate**G131**

The G131 command sets the U600 to update the servo loop 1000 times/sec. (every 1msec.). The normal process of sampling the encoder feedback and updating the current commands 4000 times per second will be reduced. When switching from 4K to 1K, the user must adjust the gains as shown below.

$$\begin{aligned}K_p &= K_p / 4 \\PGAIN &= PGAIN * 4 \\K_i &= K_i\end{aligned}$$

The default mode of operation on power-up may be set in the CNC Initialization screen for each CNC under the G-codes pull down menu.



CHAPTER 3: M-CODES**In This Section:**

- Description 3-1
- M-code Programming..... 3-5
- Assigning Virtual I/O 3-5
- The MODSCAN.INI File..... 3-8
- Programmable Logic Controllers 3-9
- PCUDIO Digital I/O Cards 3-11
- XYCOM Digital I/O Cards 3-13
- Define XYCOM I/O Board..... 3-13
- Associating Virtual I/O with Xycom I/O 3-14
- The MCODEx.INI File 3-15
- Automatically Initializing Virtual I/O 3-25
- The FAULTMSG.INI File..... 3-25

3.1. Description

In general, M-codes are used to perform miscellaneous I/O functions from within parts programs. The functions associated with each M-code vary significantly. However, in most cases, they perform one of the following functions:

- Alter parts program flow
- Set or read general purpose outputs
- Read general purpose inputs.

Table 3-1 is a list of the M-codes that are pre-defined for the UNIDEX 631/U600.

Table 3-1. M-code Summary

M-code	Page	Description
M0	3-3	Program Stop
M01	3-3	Optional Program Stop
M02	3-3	End of Program
M03	3-3	Spindle On Clockwise
M04	3-3	Spindle On Counter-clockwise
M05	3-3	Spindle Off
M19	3-3	Spindle Off/Reorient
M30	3-4	Reset to Beginning of First Program and Wait for Cycle Start
M47	3-4	Restart Program Execution
M48	3-4	Feedrate Override Lock
M49	3-4	Feedrate Override Unlock
M50	3-4	Spindle Feedrate Override Lock
M51	3-4	Spindle Feedrate Override Unlock

NOTE:

Designation of CNC specific M-codes are in the file MCODE.INI. This file provides a mechanism for associating M-codes with binary inputs, binary outputs, PLC register inputs and PLC register outputs. If you purchased the multi-CNC version of MAINMENU, you may also designate CNC specific M-codes in the file MCODEx.INI (where x is the CNC # 1 through 4).

3.1.1. Program Stop **M0**

Upon completion of all other commands in the program block, program execution will stop. You may continue program execution with the Cycle Start Control located on the CNC Run Screen (refer to the *UNIDEX 631/U600 User's Manual*)

3.1.2. Optional Stop **M01**

The functionality of this command depends upon the current state of the Optional Stop Control located on the CNC Run Screen (refer to the *UNIDEX 631/U600 User's Manual*). If this control is active, the CNC will respond to this M-code as described for the M0 command. Otherwise, this M-code is ignored.

Use of this command permits you to interrupt program execution at a specific point when an abnormal condition has occurred, but does not require operator intervention under normal circumstances.

3.1.3. End Of Program **M02**

This command ends the program.

3.1.4. Spindle On Clockwise **M03**

This command causes the spindle to begin rotating clockwise (CW). The speed of the spindle is determined by the S keyword, as well as the current operational mode or MDI mode in respect to the Spindle Speed G-code group.

3.1.5. Spindle On Counterclockwise **M04**

This command causes the spindle to begin rotating counter-clockwise (CCW). The speed of the spindle is determined by the S keyword, as well as the current operational mode or MDI mode in respect to the Spindle Speed G-Code group.

3.1.6. Spindle Off **M05**

This command causes the spindle to stop moving when this command is encountered. The spindle stops moving by being disabled.

3.1.7. Spindle Off/Reorient **M19**

This command causes the spindle to move to the zero position at the current S feedrate or spindle axes' Rapid Feedrate, if S = 0. This command should only be used for servo driven spindle axes.

3.1.8. Restart Program Execution and Wait for Cycle Start M30

Program execution immediately returns to the first line of the first program (if this is encountered in a nested, called program, execution begins at the top of the first program and all nested calls, while loops, etc., are canceled) and awaits a cycle start command to resume program execution. All modal information remains unchanged.

3.1.9. Restart Program Execution M47

Program execution immediately returns to the first line of the first program (if this is encountered in a nested, called program, execution begins at the top of the first program, and all nested calls, while loops, etc., are canceled) and begins execution from there. All modal information remains unchanged.

3.1.10. Feedrate Override Lock M48

This command disables usage of the feedrate override controls located within the CNC Manual Data Input and Run Mode screens. (Refer to the *UNIDEX 631/U600 User's Manual*.)

3.1.11. Feedrate Override Unlock M49

This command enables usage of the feedrate override controls located within the CNC Manual Data Input and Run Mode screens. (Refer to the *UNIDEX 631/U600 User's Manual*.) It does not require that the M48 command had been previously executed.

3.1.12. Spindle Feedrate Override Lock M50

This command disables usage of the spindle feedrate override controls located within the CNC Manual Data Input and Run Mode screens. (Refer to the *UNIDEX 631/U600 User's Manual*.)

3.1.13. Spindle Feedrate Override Unlock M51

This command enables usage of the spindle feedrate override controls located within the CNC Manual Data Input and Run Mode screens. (Refer to the *UNIDEX 631/U600 User's Manual*.) It does not require that the M50 command had been previously executed.

3.2. M-code Programming

M-codes perform miscellaneous I/O functions from within parts programs.

See the WTCH command for information on controlling a virtual I/O point based upon an axis' position.



The UNIDEX 631/U600 CNC provides several pre-defined functions (see the preceding section for a description of the pre-defined M-codes) and permits you to define others. In general, user defined M functions are permitted to read and write the state of user I/O.

All M-codes operate in one of two ways. Functions which do not return a value may be invoked simply by specifying the M keyword and the number of the function to be executed. M functions which return information must be followed by a "\$" and the name of the variable which is to receive the data returned. Examples of these two forms are shown below.

M0 or M200 \$VAR1

3.3. Assigning Virtual I/O

The UNIDEX 631/U600 CNC M functions may be configured by you. Functions of this type are limited to performing operations on various input and output devices present in the system.

The user defined I/O system provided by the CNC gives a generic interface to all I/O through the use of M functions. These M functions operate on a region of memory which mirrors the state of all I/O present in the system. This area of memory is constantly updated to ensure that it reflects the current state of all system inputs and outputs.

In the context of the UNIDEX 631/U600 CNC, this system of memory mapped I/O is referred to as virtual I/O. Each memory location which holds the mirror image of the actual I/O is called a virtual I/O bit. The process which monitors these bits is referred to as the modscan thread.

The system uses the file MODSCAN.INI in the \U31 directory to determine the type of I/O devices which are present in the system. If this file does not exist, the modscan thread will terminate and no user defined M functions will be supported by the UNIDEX 631. However, the UNIDEX 600 does not require a definition of its on-board I/O (16 inputs/outputs) within the MODSCAN.INI file. They are preassigned to virtual inputs/outputs 0 through 15.

Each line of this file contains information relevant to a particular device in the system. Comments may be placed in the file by preceding them with a semi-colon. The remainder of the line is assumed to be a comment. Blank lines are ignored.

The system uses the file MCODEx.INI (see note on page 3-2) to define the action of a user defined M-code (virtual I/O) and assigns it to a particular user I/O bit. Assigning a

virtual I/O bit requires two steps: an entry in the MODSCAN.INI file, and an entry in the MCODEx.INI file.

The UNIDEX 631/U600 CNC permits user defined I/O on devices of three different types; PCDIO, Xycom 32 bit digital I/O cards and Modicon Programmable Logic Controllers. These keywords, PCDIO, XYCOM and PLC refer to devices of the types found on the following pages.



Binary I/O refers to I/O bit operations.



No I/O should be mapped into the virtual outputs at 448 to 512, this is preassigned to CNCSTAT1-CNCSTAT4. Refer to Chapter 1: Axes Designators.

In the U600 systems (excluding the U631), the 16 onboard inputs and outputs are mapped into virtual inputs (0-15) and virtual outputs (0-15). They should not have any other I/O mapped into these locations, refer to Table 3-2. Table 3-2 shows that the encoder expansion board will have its I/O preassigned to specific virtual I/O values depending on the board number the encoder card is configured as.

Table 3-2. Virtual I/O Inputs and Outputs

INPUTS			
Board	Label	Virtual Input #	Connector and Pin Numbers on the Respective Board
U600	IN0-15	0 through 15	P9 pins 31 - 1 (odd pins)
Expansion Board 1	IN0-15	16 through 31	P9 pins 31 - 1 (odd pins)
Expansion Board 1	IN16-39	32 through 55	P8 pins 47 - 1 (odd pins)
Expansion Board 2	IN0-15	56 through 71	P9 pins 31 - 1 (odd pins)
Expansion Board 2	IN16-39	72 through 95	P8 pins 47 - 1 (odd pins)
Expansion Board 3	IN0-15	96 through 111	P9 pins 31 - 1 (odd pins)
Expansion Board 3	IN16-39	112 through 135	P8 pins 47 - 1 (odd pins)
OUTPUTS			
U600	OUT0-7	0 through 7	P9 pins 47 - 33 (odd pins)
U600	OUT8-15	8 through 15	P10 pins 47 - 33 (odd pins)
Expansion Board 1	OUT0-7	16 through 23	P9 pins 47 - 33 (odd pins)
Expansion Board 1	OUT8-15	24 through 31	P10 pins 47 - 33 (odd pins)
Expansion Board 1	OUT16-39	32 through 55	P7 pins 47 - 1 (odd pins)
Expansion Board 2	OUT-7	56 through 63	P9 pins 47 - 33 (odd pins)
Expansion Board 2	OUT8-15	64 through 71	P10 pins 47 - 33 (odd pins)
Expansion Board 2	OUT16-39	72 through 95	P7 pins 47 - 1 (odd pins)
Expansion Board 3	OUT0-7	96 through 103	P9 pins 47 - 33 (odd pins)
Expansion Board 3	OUT8-15	104 through 111	P10 pins 47 - 33 (odd pins)
Expansion Board 3	OUT16-39	112 through 135	P7 pins 47 - 1 (odd pins)

3.3.1. The MODSCAN.INI File

The MODSCAN.INI file allows the user to define the type of hardware present (XYCOM, PLC or PCDIO) and define each as inputs or outputs. Shown below in figure 3-1, is a typical MODSCAN.INI file.

```

; Filename : MODSCAN.INI
;-----
; This file, and the accompanying MCODEX.INI, are used to enable M functions
; to operate upon the XYCOM and PCDIO digital I/O card. The user must first specify
; the configuration of each card in the system, and then assign Virtual
; inputs and outputs to each of the real inputs And outputs.
;
; NOTE: The number of bytes specified as inputs in the configuration line
; must correspond with the number of bytes of virtual inputs allocated.
; Similarly, the number of output bytes and the number of virtual
; outputs allocated must agree.
;
; These bits may then be assigned to M functions using the MCODE.INI File
;-----
XYCOM 1 0000F400 1111 ;XYCOM Board #1 - VME A16 Address 0xF400
XYCOM 2 0000EC00 0000 ;XYCOM Board #2 - VME A16 Address 0xEC00
PCDIO 1 0000300 24 ;Configure Board 1 as a PCDIO-24P at address 0x300
PCDIO PORT 1 1111 O ;Set Board 1-Port 3 Chan A..B (Bits 0..15) as Input
;Set Board 1-Port 3 Chan C (Bits 16..23) as Output

PLC 1 11 22 33 44 55 ; plc 1

; Two Inputs Bytes
; Two Output Bytes
BI XYCOM 1 0 0 4 ;BI--> Binary Inputs associated with
;XYCOM Board #1
; Starting Bit Number 0
; Starting Virtual Input Number 0
; 4 Bytes - Virtual Inputs (32 Bits)
BO XYCOM 2 0 0 4 ;BO--> Binary Outputs associated with
;XYCOM Board #2
; Starting Bit Number 0
; Starting Virtual Output Number 0
; 4 Bytes - Virtual Outputs(32 Bits)
BI PCDIO 1 0 32 2 ;Binary Inputs associated with PCDIO Board 1
;Starting Input Bit 0
;Starting Virtual Input Bit 32
;2 Bytes of Input (16 Bits)
BO PCDIO 1 16 32 1 ;Binary Outputs associated with PCDIO Board 1
;Starting Output Bit 16
;Starting Virtual Output Bit 32
;1 Byte of Output (8 Bits)

BI PLC 1 40001 32 2
BO PLC 1 40012 32 2
RI PLC 1 40023 0 2
RO PLC 1 40034 0 2

```

Figure 3-1. MODSCAN.INI File

3.3.2. Programmable Logic Controllers

PLC

To specify the existence of a Modicon PLC in the system, you must place the keyword PLC at the beginning of a line in the MODSCAN.INI file. You must also specify several parameters relevant to accessing the device over the Modbus Plus network.

SYNTAX: *PLC <PLC#> <routing information>*

The PLC# parameter specifies the number to be used when referring to this PLC. The valid range of these numbers is 1-7. These numbers do not have to be specified sequentially.

The routing information specified is used when attempting to access the PLC over the Modbus Plus network. This parameter is composed of exactly five routing addresses, each of which has a valid range of 0-63.

Refer to your PLC documentation (MODICON PLC User's Manual) for additional information.

EXAMPLE:

PLC 6 11 22 33 44 55 ;Specifies that PLC #6 is present in the system and can be accessed using ;11 22 33 44 55 as routing addresses.

3.3.3. Associating Virtual I/O with PLC I/O

The modscan thread ensures that the state of the virtual I/O bits are consistent with the current state of the hardware. In order to perform this function, an association must be made between the virtual I/O bits and the various PLC registers. This association is made in groups and is performed as described below.

Two types of inputs and outputs are supported in relation to a PLC: binary (bit oriented) and register (byte oriented). Both of these types may be either inputs or outputs. The naming conventions shown below must be used to designate the type of I/O found in the PLC:

Bit	Type of I/O Description
BI	Binary Input Bit
BO	Binary Output Bit

Byte	Type of I/O Description
RI	Register Input
RO	Register Output

Once you have specified the type of I/O present, you must then associate this group with a particular set of 4xxxx registers resident within a particular PLC. These registers must then be associated with a specific set of virtual I/O bits.

The keyword **PLC** is used to specify that a particular set of binary inputs/outputs or register inputs/outputs are associated with a PLC. The next parameter specifies which PLC, by referencing the number defined previously using the **PLC** command.

The next step in defining a set of PLC binary/register I/O points is to associate a particular set of 4xxxx registers with a particular set of virtual I/O bits. This is done by specifying the starting 4xxxx register, the starting virtual I/O bit number and the size of each set, as parameters to this command. Valid PLC register numbers range from 40001 through 44000. Also, binary inputs may be associated with 0xxxx and 1xxxxx PLC registers.

Refer to your PLC documentation (MODICON PLC User's Manual) for additional information.

SYNTAX:

```
[BI|BO|RI|RO] PLC <#> [GLOBAL] <StartReg> <StartVirtIO> <size>
```



The keyword **GLOBAL** may also be used to reference the global data being transferred to/from the PLC. This keyword should be inserted just following the PLC number specification.

EXAMPLE:

BI PLC 1 40001 0 8	;Associate 128 binary inputs (8*16) at PLC #1 register ;40001. Starting with virtual input point #0.
BO PLC 2 41001 256 2	;Associate 32 binary outputs (2*16) at PLC#2 register ;41001. Starting with virtual output point #256.
RI PLC 7 42001 256 16	;Associate 16 register inputs with PLC #7 register ;42001 starting with virtual I/O point #256
RO PLC 3 43001 272 16	;Associate 16 register outputs with PLC #3 register ;43001 and virtual output # 272

For binary inputs and outputs, the starting virtual I/O number must be divisible by 16. Size indicates the “#” of consecutive PLC registers and must not exceed the number of virtual I/O points available.

Binary inputs may also be associated with 0xxxx and 1xxxx PLC registers.

A maximum of 512 discrete inputs, 512 discrete outputs, 128 register inputs and 128 register outputs can be defined within the MODSCAN.INI file.

The keyword GLOBAL may also be used to reference the global data being transferred to/from the PLC. This keyword should be inserted just following the PLC number specification.



3.3.4. PCDIO Digital I/O Cards

To specify the existence of a PCDIO digital I/O card in the system, you must place the keyword PCDIO (along with several other parameters) at the beginning of a line in the modscan.ini file.

3.3.4.1. Define PCDIO I/O Board

To specify a PCDIO Card in the Modscan.ini, you must configure the PCDIO cards base address and how the IO Ports are configured. See the *Industrial Computer Source, Model PCDIO Series Product Manual*, for information regarding address setup and port configurations.

Modscan.ini Syntax:

PCDIO <Board #> <Addr> <Type>

where

<Board #> is a value 1..4
 <Addr> is the address settings of the PCDIO card (in HEX)
 <Type> is the PCDIO Type (24, 48, 72, 120, 216)

PCDIO PORT <Board #> <Port #> <Ch A I/O> <Ch B I/O> <Ch C I/O>

where

<Board #> is a value 1..4
 <Port #> is the Port value 1..9 (Block of 24 Bits of data). This value is dependent on the type that is set. A PCDIO-24P has 1 Port a PCDIO-216P has 9 Ports.
 <Ch A>, <Ch B>, <Ch C> is either I or O and specifies whether the given channel for the port is to be configured for Input or output.

EXAMPLE:

Configure a PCDIO-120P for 64 bits of input and 56 bits of output.

PCDIO 1 0000200 120	;Configure Board 1 as a PCDIO 120P at address 0x200
PCDIO PORT 1 1 I I I	;Set Board 1 Port 1 Chan A..C (Bits 0..23) as Input
PCDIO PORT 1 2 I I I	;Set Board 1 Port 2 Chan A..C (Bits 24..47) as Input
PCDIO PORT 1 3 I I O	;Set Board 1 Port 3 Chan A..B (Bits 48..63) as Input
	;Set Board 1 Port 3 Chan C (Bits 64..71) as Output
PCDIO PORT 1 4 O O O	;Set Board 1 Port 4 Chan A..C (Bits 72..95) as Output
PCDIO PORT 1 5 O O O	;Set Board 1 Port 5 Chan A..C (Bits 96..119) as Output
BI PCDIO 1 0 0 8	;Binary Inputs associated with PCDIO Board 1
	;Starting Input Bit 0
	;Starting Virtual Input Bit 0
	;8 Bytes of Input (64 Bits)
BO PCDIO 1 48 0 7	;Binary Outputs associated with PCDIO Board 1
	;Starting Output Bit 48
	;Starting Virtual Output Bit 0
	;7 Bytes of Output (56 Bits)

Mcode.ini Syntax for PCDIO:

Binary Input M-Code Initialization

MXXXX BI PCDIO <Brd #> <Bit #>

Binary Output M-Code Initialization

MXXXX BO PCDIO <Brd #> <Bit #> <Level>

3.3.5. XYCOM Digital I/O Cards

XYCOM

To specify the existence of a Xycom digital I/O card in the system, you must place the keyword XYCOM at the beginning of a line in the MODSCAN.INI file. You must also specify several parameters relevant to the configuration of the XYCOM digital I/O card.

3.3.6. Define XYCOM I/O Board

A board number parameter is used when first defining a particular XYCOM card. You may then refer to XYCOM board #x and the system can determine any information necessary.

The VME address parameter specifies the base address of the XYCOM digital I/O card on the VME backplane. This address is configured via jumpers on the board (consult the *XYCOM Digital I/O Card User's Manual* for the locations and settings of these jumpers). Valid base addresses for XYCOM I/O cards are F400, EC00, E800 and E400.

The XYCOM digital I/O card has 32 bits of I/O associated with it. The UNIDEX 631/U600 CNC treats these bits as 4 groups of 8 bits each. Four parameters are used to configure the functionality of each of these four groups.

Valid values for each of these parameters are I (input) and O (output). Although 0-4 groups may be configured as inputs, all of these must be defined before defining the outputs. That is, input bytes must be configured in a contiguous block and located before any output blocks in the specification of a single XYCOM card.

SYNTAX: XYCOM <board_num> <VME address> [I/O] [I/O] [I/O] [I/O]

EXAMPLES:

XYCOM 1 0000F400 I I O O	;Xycom Board #1, at VME address 0xF400. ;2 Bytes are Inputs and 2 Bytes are Outputs.
XYCOM 2 0000EC00 I I I O	;Xycom Board #2, at VME address 0xEC00. ;3 Bytes are Inputs and 1 Bytes is Output.



Although 0-4 groups may be configured as inputs, all of these must be defined before defining the outputs. That is, input bytes must be configured in a contiguous block and located before any output blocks in the specification of a single XYCOM card.

3.3.7. Associating Virtual I/O with Xycom I/O

The modscan thread ensures that the state of each virtual I/O bit is consistent with the state of the hardware. In order to perform this function, an association must be made between the virtual I/O bits and the various Xycom digital I/O bits. This association is made in groups and is performed as described below.

The type of I/O supported for Xycom cards is defined as Binary Inputs (BI) and Binary Outputs (BO). Therefore, either of these two keywords may be used to designate an I/O group. Once this type of I/O group has been defined, the individual XYCOM bits must be associated with specific virtual I/O bits. The following describes how this is accomplished.

The keyword XYCOM is used to specify that a particular set of Binary Inputs/Outputs is associated with a XYCOM card. The next parameter specifies the card number. This XYCOM card number must be pre-defined using the XYCOM command described previously.

You must specify the starting bit number on the Xycom Card, which is associated with a group. This bit number must be less than 32 and evenly divisible by 8. The starting virtual I/O point parameter, specified next, must also be evenly divisible by 8. The valid range for virtual I/O points is currently 0-511. The final parameter, size, specifies the number of 8 bit groups which are to be included in this group. This value has a valid range of 1-4.

SYNTAX: *[BI|BO] XYCOM <#> <StartXycomBit> <StartVirtIO> <size>*

EXAMPLE:

BI XYCOM 1 0 0 2	;The following assigns XYCOM Bits 0-15 as ;virtual inputs 0-15. ;BI--> Binary Inputs associated with ;XYCOM Board #1. ;Starting XYCOM Bit Number 0. ;Starting Virtual Input Number 0. ;2 Bytes - Virtual Inputs (16 Bits).
BO XYCOM 1 16 0 2	;The following assigns XYCOM Bits 16-31 ;as virtual outputs 0-15. ;BO--> Binary Outputs associated with ;XYCOM Board #1. ;Starting XYCOM Bit Number 16. ;Starting Virtual Output Number 0. ;2 Bytes - Virtual Outputs (16 Bits).

3.3.8. The MCODEx.INI File

The file MCODEx.INI associates user defined M functions with specific virtual I/O points. The file MCODE1.INI is used for CNC engine #1, MCODE2.INI is associated with CNC engine #2, etc.

Each line in the MCODEx.INI file contains information for initializing a particular M-code. Comments can be placed in the file by inserting a semi-colon as the first character of the line. Blank lines are ignored.

There are four types of M-codes available. The first two types are binary input and binary output (both bit oriented). The last two types are register input and register output (both byte oriented). Each of these are described in the following sections.

Figure 3-2, shows a typical MCODEx.INI file.

```
; Filename: MCODEx.INI
;-----
; This file and the accompanying MODSCAN.INI file are used to initialize
; the following M-Codes for the U31/U600 CNC
;
; MCODEs 1020 - 1035 Assert XYCOM Board #1 Outputs 16-31
;       1040 - 1055 De-assert XYCOM Board #1 Outputs 16-31
;       1060 - 1075 Assert XYCOM Board #2 Outputs 16-31
;       1080 - 1095 De-assert XYCOM Board #2 Outputs 16-31
;       3000 - 3015 Associate with PCDIO Board #1 Inputs 0-15
;       3050 - 3057 Assert PCDIO Board #1 Outputs 16-23
;       3050 - 3057 De-assert PCDIO Board #1 Outputs 16-23
;       4000-4012 to use U600 onboard I/O
; NOTE: The MODSCAN.INI file must contain configuration information
;       for all XYCOM and PCDIO cards. It must also assign all the
;       virtual I/O Bits for the PCDIO and XYCOM I/O Bits
;-----
M1020 BO XYCOM 1 16 L1      ; Mxxxx Binary Output XYCOM board #1 Bit #x
M1021 BO XYCOM 1 17 L1      ;                               Logic Level 1
M1022 BO XYCOM 1 18 L1
M1023 BO XYCOM 1 19 L1
M1024 BO XYCOM 1 20 L1
M1025 BO XYCOM 1 21 L1
M1026 BO XYCOM 1 22 L1
M1027 BO XYCOM 1 23 L1
M1028 BO XYCOM 1 24 L1
M1029 BO XYCOM 1 25 L1
M1030 BO XYCOM 1 26 L1
M1031 BO XYCOM 1 27 L1
M1032 BO XYCOM 1 28 L1
M1033 BO XYCOM 1 29 L1
M1034 BO XYCOM 1 30 L1
M1035 BO XYCOM 1 31 L1
;-----
M1040 BO XYCOM 1 16 L0      ; Mxxxx Binary Output XYCOM board #1 Bit #x
M1041 BO XYCOM 1 17 L0      ;                               Logic Level 0
M1042 BO XYCOM 1 18 L0
```

```
M1043 BO XYCOM 1 19 L0
M1044 BO XYCOM 1 20 L0
M1045 BO XYCOM 1 21 L0
M1046 BO XYCOM 1 22 L0
M1047 BO XYCOM 1 23 L0
M1048 BO XYCOM 1 24 L0
M1049 BO XYCOM 1 25 L0
M1050 BO XYCOM 1 26 L0
M1051 BO XYCOM 1 27 L0
M1052 BO XYCOM 1 28 L0
M1053 BO XYCOM 1 29 L0
M1054 BO XYCOM 1 30 L0
M1055 BO XYCOM 1 31 L0
;-----
M1060 BO XYCOM 2 16 L1 ; Mxxxx Binary Output XYCOM board #1 Bit #x
M1061 BO XYCOM 2 17 L1 ;                               Logic Level 1
M1062 BO XYCOM 2 18 L1
M1063 BO XYCOM 2 19 L1
M1064 BO XYCOM 2 20 L1
M1065 BO XYCOM 2 21 L1
M1066 BO XYCOM 2 22 L1
M1067 BO XYCOM 2 23 L1
M1068 BO XYCOM 2 24 L1
M1069 BO XYCOM 2 25 L1
M1070 BO XYCOM 2 26 L1
M1071 BO XYCOM 2 27 L1
M1072 BO XYCOM 2 28 L1
M1073 BO XYCOM 2 29 L1
M1074 BO XYCOM 2 30 L1
M1075 BO XYCOM 2 31 L1
;-----
M1080 BO XYCOM 2 16 L0 ; Mxxxx Binary Output XYCOM board #1 Bit #x
M1081 BO XYCOM 2 17 L0 ;                               Logic Level 0
M1082 BO XYCOM 2 18 L0
M1083 BO XYCOM 2 19 L0
M1084 BO XYCOM 2 20 L0
M1085 BO XYCOM 2 21 L0
M1086 BO XYCOM 2 22 L0
M1087 BO XYCOM 2 23 L0
M1088 BO XYCOM 2 24 L0
M1089 BO XYCOM 2 25 L0
M1090 BO XYCOM 2 26 L0
M1091 BO XYCOM 2 27 L0
M1092 BO XYCOM 2 28 L0
M1093 BO XYCOM 2 29 L0
M1094 BO XYCOM 2 30 L0
M1095 BO XYCOM 2 31 L0
;-----
;Mxxxx Binary Input PCDIO board # 1 Bit # x
M3000 BI PCDIO 1 0
M3001 BI PCDIO 1 1
M3002 BI PCDIO 1 2
M3003 BI PCDIO 1 3
M3004 BI PCDIO 1 4
M3005 BI PCDIO 1 5
M3006 BI PCDIO 1 6
M3007 BI PCDIO 1 7
```

```

M3008 BI PCDIO 1 8
M3009 BI PCDIO 1 9
M3010 BI PCDIO 1 10
M3011 BI PCDIO 1 11
M3012 BI PCDIO 1 12
M3013 BI PCDIO 1 13
M3014 BI PCDIO 1 14
M3015 BI PCDIO 1 15
:.....
;Mxxxx Binary Output PCDIO board # Bit #
;Logic Level 1
M3050 BO PCDIO 1 16 L1
M3051 BO PCDIO 1 17 L1
M3052 BO PCDIO 1 18 L1
M3053 BO PCDIO 1 19 L1
M3054 BO PCDIO 1 20 L1
M3055 BO PCDIO 1 21 L1
M3056 BO PCDIO 1 22 L1
M3057 BO PCDIO 1 23 L1
:.....
;Mxxxx Binary Output PCDIO board # Bit #
;Logic Level 0
M3060 BO PCDIO 1 16 L0
M3061 BO PCDIO 1 17 L0
M3062 BO PCDIO 1 18 L0
M3063 BO PCDIO 1 19 L0
M3064 BO PCDIO 1 20 L0
M3065 BO PCDIO 1 21 L0
M3066 BO PCDIO 1 22 L0
M3067 BO PCDIO 1 23 L0
:.....
;To use U600 onboard inputs use the following format
M4000 BI virtual 0 ;M4000 to read Bit 0
;To use U600 onboard outputs
M4010 B0 virtual 2 ;M4010 to set/reset Bit, you must specify the
;state (1 or 0) after the mcode within the user
;program (i.e., M4010 1)
M4011 B0 virtual 2 L0 ;M4011 will set bit 2 to 0 without specifying the
;state after the mcode in the user program
M4012 B0 virtual 2 L1 ;M4012 sets bit 2 to 1

```

Figure 3-2. MCODEx.INI File

3.3.9. Binary Input M-code Initialization

SYNTAX: *Mxxxx BI <input device>*

where:

xxxx refers to the number of the M-function being defined

<input device> refers to one of the following device types:

PLC Register
 Example PLC Plc#Reg#B bit#
 PLC 1 40010 B 1

where:

Plc# PLC Number 1-7
 Reg# PLC Register from 40001 to 44000
 Bit# Register Bit Number from 1-16.

PLC Global

Example PLC Plc# GLOBAL Glob# B Bit#
 PLC 1 GLOBAL 2 B 1

where:

Plc# PLC Number 1-7
 Glob# Global Data byte to which this applies (1-32)
 Bit# Register Bit Number from 1-16.

Virtual

Example VIRTUAL Virt#
 VIRTUAL 2

where:

Virt# Virtual I/O point.

PCDIO

Example PCDIO Brd# Bit#
 PCDIO 1 1

where:

Brd# Refers to the PCDIO Board number
 Bit# Register Bit Number from 1-16.

Xycom

Example XYCOM Brd# Bit#
 XYCOM 1 1

where:

Brd# Refers to the Xycom Board Number
 Bit# Register Bit Number from 1-16.

Virtual Ouput (Reading the state of outputs)

Example Virtual Output #.

Is the virtual output the mcode will return the value of.

EXAMPLE:

M1078 BI PLC 1 40020 B 4	;M1078 returns the value of PLC#1 register
M1079 BI PLC 2 GLOBAL 3 B 1	#40020, Bit 4
M1080 BI VIRTUAL 2	;M1079 returns the value of PLC#2 GLOBAL3
M1081 BI XYCOM 1 1	;M1080 returns the value of virtual input bit 2
M1082 BI VIRTUAL OUTPUT 10	;M1081 returns the value of XYCOM bd.#1 bit 1
M1083 BI PCDIO 1 5	;M1082 return the value of virtual output bit 10
	;M1083 returns the value of PCDIO Brd# bit 5

M1082 illustrates how to read the value of a binary output with an M-code. Its usage in a program would be M1082 var, where var is the variable that will return the value of the output bit.



3.3.10. Binary Output M-code Initialization

SYNTAX: *Mxxxx BO <output device> <options>*

where:

xxxx refers to the number of the M-function being defined
<Output device> refers to one of the following device types:

PLC Register

PLC Plc#Reg#B Bit# Level

Example PLC 1 40010 B 1 L0

where:

Plc# PLC Number 1-7

Reg# PLC Register from 40001 to 44000

Bit# Refers to the particular bit on that board

Level 0 clears Output, 1 sets Output.

PLC Global

PLC Plc# GLOBAL Glob# B Bit# Level

Example PLC 1 GLOBAL 2 B 1 L0

where:

Plc# PLC Number 1-7

Glob# Global Data byte to which this applies

Bit# Refers to the particular bit on that board

Level 0 clears Output, 1 sets Output.

Virtual

VIRTUAL Virt# Level

Example VIRTUAL 2 L0

PCDIO

PCDIO Brd# Bit# Level

Example PCDIO 1 9 L1

Xycom

XYCOM Brd# Bit# Level

Example XYCOM 1 9 L1

The available options for initializing binary outputs are described below in the order in which they must appear. Any options which are omitted will use default values.

- <L0,L1> This option specifies the output to be either set (1) or cleared (0). If this option is not specified, then the output level must be specified within the parts program.
- <FB [feedback_device]> This option specifies a feedback input to be associated with the binary output. The [feedback_device] must be one of the following formats: PLC Register, PLC Global, VIRTUAL or XYCOM. Using this option causes the binary output to be cleared\set until the feedback device responds. When this occurs the binary output is set\cleared. The default is no associated feedback device.
- <FLT [fault_device]> This option specifies a fault feedback to be associated with the binary output. The [fault_device] must be one of the following formats: PLC Register, PLC Global, VIRTUAL or XYCOM. The default is no associated fault device.

This can only be used if the <FB> option is specified.
- <BEFORE,AFTER> This option specifies whether the binary output is cleared\set before or after the completion of the program block. The default is AFTER.
- <RESET,NOWAITRESET> When a feedback device is specified along with the reset command, the binary output remains active until the feedback is active. Then the binary output will go inactive. The next line in the program will not execute until the feedback becomes inactive. If "NOWAITRESET" is specified, the next line of the program executes without waiting for the feedback to go active.
- <TIMEOUT time> When feedback is specified with timeout, the CNC will wait **time** (time specified by the user in milliseconds) for the feedback to become active after setting the binary output active. If the feedback fails to become active during the time specified, then a CNC fault will be generated.

EXAMPLE:

M1065 BO XYCOM 1 24	;M1065 1" sets XYCOM bd.#1 bit 24, "M1065 0" would clear it
M1066 BO PLC 1 40020 B 3 L1	;M1066 sets PLC#1 register #40020 bit 3 to logic 1
M1067 BO PLC 2 GLOBAL 4 B 4 L0	;M1067 sets PLC#2 GLOBAL4 , bit 4 to logic 0
M1068 BO VIRTUAL 5 L1	;M1068 sets VIRTUAL bit5 to logic 1
M1069 BO XYCOM 1 25 L1	;M1069 sets XYCOM bd.#1 bit 25 to logic 1
M1070 BO XYCOM 1 25 L0 1	;M1070 sets XYCOM bd.#1 bit 25 to logic 0
M1071 BO XYCOM 1 25 L0 FB PLC 1 40010 B 1 L0	;M1071 sets XYCOM bd#1 bit 25 to logic 0 until ; PLC#1 register 40010 bit 1 is low
M1072 BO PCDIO 1 17 L1	;M1072 sets PCDIO Brd# 1 bit 17 to logic 1
M1073 BO PCDIO 1 17 L0	;M1073 sets PCDIO Brd# 1 bit 17 to logic 0
M1074 BO VIRTUAL 10 LI FB VIRTUAL 20 L1 TIMEOUT 100 NOWAITRESET	;M1072 sets virtual output 10 high until virtual input 20 becomes ;high or 100 millisec the set output 10 low

3.3.11. Register Input M-code Initialization

SYNTAX: *Mxxxx RI <input device>*

where:

xxxx refers to the number of the M-function being defined

<input device> refers to one of the following device types:

PLC Register

PLC Plc#Reg#

Example PLC 1 4010

where:

Plc# PLC Number from 1-7

Reg# PLC Register from 40001 to 44000.

PLC Global

PLC Plc# GLOBAL Glob#

Example PLC 1 GLOBAL 2

where:

Plc# PLC Number 1-7

Glob# Global Data byte to which this applies.

EXAMPLE:

M1040 RI PLC 1 40100	;M1040 returns the value of PLC#1 register 40100
M1041 RI PLC 1 GLOBAL 2	;M1041 returns the value of PLC#1 GLOBAL 2
M1042 RI VIRTUAL 2	;M1042 returns the value of VIRTUAL INPUT ;REGISTER 2



Xycom is not applicable in this command format.

3.3.12. Register Output M-code Initialization

SYNTAX: *Mxxxx RO [output device] <options>*

where:

xxxx refers to the number of the M-function being defined

<Output device> refers to one of the following device types:

PLC Register

PLC Plc#Reg#

Example PLC 1 40010

where:

Plc# PLC Number 1-7

Reg# PLC Register from 40001 to 44000.

PLC Global

PLC Plc# GLOBAL Glob#

Example PLC 1 GLOBAL 2

where:

Plc# PLC Number 1-7

Glob# Global Data byte to which this applies.

Xycom is not applicable in this command format.



<OPTIONS>

The available options for initializing register outputs are described below in the order in which they must appear. Options may be left out, in which case the default values will be used.

<FB [feedback_device]>

This option specifies a feedback to be associated with the register output. The [feedback_device] must be one of the following formats: PLC Register, PLC Global, VIRTUAL or XYCOM. Using this option causes the register output to be set until the feedback device responds. When this occurs, the register output is reset. The default is no associated feedback device.

An additional option is available after a feedback designation. This can only be used if the <FB> option is specified.

<FLT [fault_device]>

This option specifies a fault feedback to be associated with the register output. The [fault_device] must be one of the following formats: PLC Register, PLC Global, VIRTUAL or XYCOM. The default is no associated fault device.

This can only be used if the <FB> option is specified.

<BEFORE,AFTER>

This option specifies whether the register output is set before or after the completion of the program block. The default is BEFORE.

EXAMPLE:

M1100 RO PLC 1 40100	;M1100 writes a WORD PLC#1 register 40100
M1110 RO PLC 2 GLOBAL 5	;M1110 writes a WORD to PLC#2 GLOBAL 5
M1120 RO VIRTUAL 3	;M1120 writes a WORD to VIRTUAL OUTPUT
	;REGISTER 3

3.4. Automatically Initializing Virtual I/O

To automatically initialize virtual I/O, create an ASCII text file in the name of VIRT.INI in the U31\INI subdirectory. The format is “Virtual_I/O_Num Value” repeating on as many lines as required to initialize the I/O points.

Figure 3-3 shows a sample ASCII text file.

```
0 1
2 1
4 1
```

Figure 3-3. VIRT.INI File

This sets virtual outputs 0, 2, 4 to a logic 1.

3.5. The FAULTMSG.INI File

The FAULTMSG.INI file allows text to be associated with an error condition for any of the virtual I/O points. This text will be displayed in the fault window when the designated I/O error occurs. Using the mouse and clicking on the message in the error window will open a help window displaying more information about the error. If a filename is specified for that bit in the faultmsg.ini file.

The general format of the FAULTMSG.INI is as follows:

SYNTAX: *I/O_TYPE# BIT# FAULT_LEVEL ERROR_TEXT [,FILENAME]*

Where:

I/O_TYPE specifies the type of I/O and register number (if applicable).

I/O_TYPE may be:

- PLC# (# is 1-7)
- RI 4xxxx (4xxxx is a valid PLC register number)
- RO 4xxxx (4xxxx is a valid PLC register number)
- BI (no parameter)
- BO (no parameter)
- GLOBAL# (# is 1-32).

BIT# is a virtual bit number (0-511).

FAULT_LEVEL is the error state, either logic 0 or 1.

ERROR_TEXT is the error message to be displayed in the window.

FILENAME is an optional filename that will provide more information if the error message in the error window is clicked on with the mouse. The specified file is an ascii text file containing information on the error message. Each I/O error must have a separate help file.

EXAMPLES:

<i>IO_TYPE#</i>	<i>BIT#</i>	<i>FAULT_LEVEL</i>	<i>ERROR_TEXT</i>	<i>[,FILENAME]</i>
PLC 2	1	0	plc2 error on bit 1	,c:\u31\ini\plc.hlp
RI 40100	3	1	reg. 40100 in bit 3 error	,c:\u31\register.hlp
RO 40110	5	0	reg. 40110 out bit 5 error	
BI	7	0	binary in bit 7 error	,c:\u31\program\binary.hlp
BO	9	1	binary output 9 failed	
GLOBAL 3	11	1	global 3 bit 11 error	

▽ ▽ ▽

CHAPTER 4: EXTENDED COMMANDS

In This Section:	
• Description	4-1
• Defining Commands.....	4-5
• Programming Operators	4-10
• Relational Operators.....	4-15
• Commands which Affect Program Flow	4-17
• Custom Display Window Commands.....	4-28
• Synchronous Motion Commands	4-34
• User Stack Operations.....	4-37
• Miscellaneous Extended Commands.....	4-41
• File Operation Commands.....	4-53
• Master-slave Motion Commands.....	4-56
• Asynchronous Motion Commands	4-59

4.1. Description

In addition to G-code and M-code programming, the UNIDEX 631/U600 CNC also provides a set of commands that permit you to control program flow and perform other miscellaneous functions. These commands are referred to as the extended command set.

This chapter contains a discussion of each extended command. The commands have been grouped into blocks of related commands. The following is a summary of all extended commands, in alphabetical order.

Table 4-1. Extended Command Summary

Extended Command	Page	Description
+	4-10	Produces a sum of two operands
-	4-10	Produces a difference of two operands
*	4-10	Produces a product of two operands
/	4-10	Produces quotient from Operand1 divided by Operand2
^	4-11	Produces the value of Operand1 raised to the power of Operand2
()	4-12	Specifies order in which to evaluate an expression
ABS	4-11	Produces the absolute value of its only operand
ACOS	4-13	Computes angle whose cosine value is the operand
AFCO	4-41	Positions an axis in relation to an optimum point
AND	4-14	Produces the logical and of two specified operands
ASIN	4-13	Computes the angle whose sine value is the operand
ATAN	4-13	Computes the angle whose tangent value is the operand

Table 4-1. Extended Command Summary Con't

Extended Command	Page	Description
CLOSECDW	4-29	Close the Custom Display Window
CLS	4-26	Call a subroutine and return upon completion
CLLS	4-26	Call a library subroutine and return when done
CONFIGM	4-57	Configures master slave relationship of axis
COS	4-13	Produces the cosine of the specified angle
DATA	4-44	Enable data collection
DENT	4-8	Define entry point
DEFINE	4-5	Define a symbol
DFS	4-8	Define subroutine
DISPLAY	4-29	Display item within the Custom Display Window
DVAR	4-6	Define variable
ENDM	4-34	Ends the motion on a single axis
EQ	4-15	True if the value of the first number equals the second number
EXECUTE	4-27	Execute a DOS or OS/2 program and return completion code
FEDM	4-59	Adds additional movement to slave
FEOF	4-53	Determines whether end of file has been reached
FILECLOSE	4-54	Closes the user data file
FILEOPEN	4-53	Opens the user data file for reading or writing
FILEREAD	4-54	Reads the data selected within the file
FILERESET	4-54	Resets the pointer within the data file to the start of the file for reading or writing
FILEWRITE	4-55	Writes the information into the selected data file
FRAC	4-11	Retrieves fraction of a floating point number
FREE	4-34	Deallocates memory that was previously allocated to a cam table
GE	4-16	True if the value of the first number exceeds or is equal to the value of the second number
GETPARM	4-48	Get the current value of an axis parameter

Table 4-1. Extended Command Summary Cont'd

Extended Command	Page	Description
GT	4-16	True if the value of the first number exceeds the value of the second number
HAND	4-34	Allows positioning of an axis with a handwheel
HARDNAMES	4-41	Enables axis designations as X/Y/Z/U/V/W/A/B/C and x/y/z/u/v/w/a
HOME/REF	4-35	Initiate homing cycle
IF-THEN-ELSE-ENDIF	4-19	Conditional execution operator
INCLUDE	4-43	Include another parts program
INDEX	4-59	Starts relative move while continuing with next program line
INT	4-11	Rounds its operand to the nearest integer value
JUMP	4-18	Jump to the specified entry point
LE	4-16	True if the value of the first number is less than or equal to the value of the second number
LT	4-16	True if the value of the first number is less than the value of the second number
MOD	4-10	Produces the remainder based on Operand1 divided by Operand2
MONSPD	4-50	Monitors speed of specified axis (when not accelerating or decelerating)
MOVETO	4-60	Moves a specified axis to a specific position at a specified speed
NE	4-15	True if the value of the first number is not equal to the value of the second number
NOT	4-14	Produces the one's complement of the binary value of its operand
OPENCW	4-28	Open Custom Display Window
ONERRGOTO	4-22	Conditional Branch On Error Conditions
OR	4-14	Produces logical OR of two specified operands
OSC	4-60	Causes a specified axis to oscillate (cycle) a specified distance and speed
POP	4-38	Remove a value from your stack
PROBE	4-48	Enables the digital touch probe
PUSH	4-37	Place a value onto your stack

Table 4-1. Extended Command Summary Con't

Extended Command	Page	Description
RECORDON	4-32	Record display messages to a file
RECORDOFF	4-33	Stop recording display messages to a file
RPT/ENDRPT	4-21	Repeat blocks specified number of times, then end
SETPARM	4-50	Set an axis parameter to the specified value
SHL	4-15	Performs a logical shift (to left) of Operand1
SHR	4-15	Performs a logical shift (to right) of Operand1
SIN	4-13	Produces the sign of the specified angle
SLEW	4-36	Allows the user to position the axis manually with a trackball or mouse
SOFTNAMES	4-41	Enables user defined axis names
SQRT	4-11	Produces the square root of its operand
STRM	4-60	begins motion on a single axis without stopping at any given target position
SYNC	4-57	Syncs (or desyncs) master-slave relationship
TAN	4-13	Produces the tangent value of the specified angle
WAIT	4-51	Will hold position until a specified condition is met
WHILE-DO-ENDWHILE	4-24	Execute blocks while condition is true
WTCH	4-52	Controls the virtual IO bit if axis position is within a specified range
XOR	4-14	Produces exclusive or of two provided operands

Commands must be enclosed within parenthesis "()". Exceptions to this rule will be noted where applicable.

4.2. Defining Commands

Some of the extended commands supported by the UNIDEX 631/U600 CNC provide you the ability to define variables, labels and subroutines. Others permit operations to be performed. The commands described in this section are used to define these.

All named variables and routines supported by the CNC have the same format. They must be at least two, but not more than twenty characters and may contain any alpha-numeric characters. However, the first three characters must be alphabetic. Upper case characters (A-Z) and lower case characters (a-z) are considered distinct (i.e., names are case-sensitive).

All variables and routines must have a unique name. That is, a variable may not have the same name as a subroutine, axis softname, etc.

All programming examples found in this section illustrate these definitions. Please refer to the examples in the next section for references on the usage within program blocks.



4.2.1. Define Symbolic Constant

DEFINE

This command gives a symbolic name to a numeric constant. This name can be referenced anywhere within the parts program. This serves to make parts programs more readable, as well as easier to maintain when the symbol is referenced multiple times.

SYNTAX: (*DEFINE Symbol Value*)
 or
 (*DEFINE Symbol = Value*)

EXAMPLE:

(DEFINE PI 3.14159)	;Define the value of PI
(DEFINE RADIUS1 = 2.0)	;Define the symbol RADIUS1 as 2.0

4.2.2. Define Local Variable**DVAR**

A variable is a location of memory which is used by a program to hold a numeric floating point value. The value of this variable may be modified from within the parts program and used in subsequent operations such as numerical calculations. Variables are also very useful for directing program flow (refer to the IF command below). Refer to section 1.4 in Chapter 1 for more details on variables.

The DVAR command defines a variable name for use within a parts program. It may then be read or written to, from within any subsequent program block in that program. Specific examples of variable usage will be included in each programming example provided throughout the following section.

SYNTAX: (*DVAR*, <variable name>, <variable name>, <variable name> ...)

EXAMPLE:

(DVAR,VAR1)	;Define a local variable named VAR1
(DVAR,DIST,SPEED)	;Define local variables named DIST and SPEED



All local variables defined are automatically initialized to zero upon program initialization.

All variables defined in this manner are available to the entire parts program. But local variables defined in one program are not accessible by other programs running on the same CNC (see static variables). Also, local variables are not accessible by programs running under other CNCs (see global variables). The UNIDEX 631/U600 CNC does not support variables local to subroutines.

DVAR commands can also be used for defining arrays.

4.2.3. Define Local Variable Arrays

DVAR

The DVAR command may also be used to define an array. An array is a group of related variables. During the definition of an array, you must specify the number of variables found in the array. Individual variables can be accessed by specifying the array name followed by a number stating which variable is to be referenced.

The number of variables within the array is typically referred to as the array size. Each individual variable is called an array element and the number stating which variable is to be accessed is referred to as the array index.

The DVAR command may be used to define arrays by following the variable/array name with an array size specification. This specification must be enclosed within brackets, "[]". When accessing the array elements, the program must also place the array index within brackets. The valid range of an array index is from zero to one less than the array size.

SYNTAX: (DVAR, <arrayname> [size])

EXAMPLE:

```
(DVAR,TSTARRAY[5])      ;Define an array, TSTARRAY, which
                        ;contains five variables. These variables can
                        ;be accessed as follows"
                        ; TSTARRAY[0]
                        ; TSTARRAY[1]
                        ; TSTARRAY[2]
                        ; TSTARRAY[3]
                        ; TSTARRAY[4]
```

A variable may also be used to hold the value of an array index (i.e. TSTARRAY[VAR1]).

Each element of the array counts as one user variable.

Refer to section 1.1.5 in Chapter 1 for details on the limitations regarding array accessing.



4.2.4. Define Entry Point**DENT**

An entry point is a label within a parts program which may be referred to by other extended commands. They are most often used to specify the destination for commands which modify program flow, such as the JUMP command (see Section 4.4.1.).

The DENT command defines an entry point (label). The name specified may then be referred to by other parts of the program.

SYNTAX: (*DENT*, <entry point name>)

EXAMPLE:

(DENT, LABEL1)	;The name LABEL1 may now be referenced from other locations within the parts program
----------------	--



All entry points are accessible from any point within the parts program. The UNIDEX 631/U600 CNC does not support entry points local to subroutines.

4.2.5. Define Subroutine**DFS**

A subroutine is a group of program blocks which may be referred to as one execution unit. Typically, this group of blocks performs some task which is to be performed multiple times within the execution of a parts program. For each point in the program that requires this task to be performed, a call can be made to the appropriate subroutine. The subroutine will be executed and upon completion, program flow will return to the point from which the subroutine was called.

The advantage of a subroutine is that it can decrease the parts program size. Multiple copies of program blocks that perform the same task can be replaced with one subroutine call. However, CNC subroutines unlike “C” language subroutines, do not accept parameters or have local variables.

The DFS command designates a group of program blocks which constitute a subroutine. As mentioned, this routine may be called from various places within the parts program, thus allowing the flow of execution to return to the calling point upon completion.

As with most extended commands, this command block must begin with a left parenthesis "(" and ends with a right parenthesis ")". However, in this case the right parenthesis specifies the end of the group of blocks included within the subroutine (see the example provided with this command). This closing parenthesis must occupy its own program block (program line).

It should be noted that subroutines do not change modal motion G-codes (G1, G2, G3) in the calling routines. For example, if the calling routine executes a G2 move then calls a subroutine that executes a G1 move, the G2 mode will be restored upon return from the subroutine.

Also, subroutines do not have a separate variable set, they use the same local variables as the program that called it.

SYNTAX: (*DFS*, <subroutine name>
 <program block>
 ...
 <program block>
)

EXAMPLE:

(DFS, MOVEHOME	;Define the subroutine movehome
G90 G0 X0. Y0.	;Move to absolute location 0,0, but note that G90 is mode
	;of operation upon completion of the subroutine
M1000	;Set output to show we're home
G04 F1.	;Wait 1 sec for other equipment to see output
)	;End the subroutine

This command must occupy its own block within a program.

Subroutines may call other subroutines, the maximum level of calling depth allowed is 20.



4.3. Programming Operators

The UNIDEX 631/U600 CNC supports a number of arithmetic, trigonometric, logical, and relational operators that permit the user to perform various calculations during program execution. Each of these operators will be discussed in the following sections.

4.3.1. Arithmetic Operators

The arithmetic operators supported by the UNIDEX 631/U600 permit the user to perform calculations during program execution. The following sections give examples of their usage.

4.3.1.1. Addition +

The + operator produces the sum of the two operands specified.

EXAMPLE: $VAR1 = VAR2 + 1.0$

4.3.1.2. Subtraction -

The - operator produces the difference between the two operands specified. This difference is computed by subtracting Operand2 from Operand1.

EXAMPLE: $VAR1 = VAR2 - 1.0$

4.3.1.3. Multiplication *

The * operator produces the product of two operands specified.

EXAMPLE: $VAR1 = 2.0 * VAR2$

4.3.1.4. Division /

The / operator assumes that Operand1 is the dividend and Operand2 is the divisor. The result is the quotient produced when dividing these two operators.

EXAMPLE: $VAR1 = VAR2 / 2.0$

4.3.1.5. Modulus MOD

This operator assumes that Operand1 is the dividend and Operand2 is the divisor. The result is the remainder produced when dividing the following two operators. For example, 1.6 equals 5.6 MOD 2.

EXAMPLE: $VAR1 = VAR2 MOD 2.0$

$VAR1 = VAR1 MOD VAR2$

4.3.1.6. Exponentiation**^**

The ^ produces the value of Operand1 raised to the power of Operand2.

EXAMPLE: VAR1 = 2 ^ VAR2
 VAR1 = VAR1 ^ VAR2

4.3.1.7. Absolute Value**ABS**

This operator produces the absolute value of its only operand. The result is always positive.

EXAMPLE: VAR1 = ABS(VAR2)
 VAR1 = ABS(-2.0)

4.3.1.8. Truncation**INT**

The INT operator returns the integer portion of the number. For example, INT(1.9) = 1. Note that INT rounds up negative numbers, for example INT (-1.1) = -2.

EXAMPLE: VAR1 = INT(VAR2)
 VAR1 = INT(1.2345)
 1 = INT(1.2)
 1 = INT(1.7)
 -2 = INT(-1.2)
 -2 = (INT(-1.7))

4.3.1.9. Fractional**FRAC**

The FRAC operator retrieves the fractional portion of a floating point number. The sign of the result is the same as that of the operand.

EXAMPLE: VAR1 = FRAC(VAR2)
 VAR1 = FRAC(6.789)

4.3.1.10. Square Root**SQRT**

The SQRT operator produces the square root of its operand. This operand must be a positive number.

EXAMPLE: VAR1 = SQRT(VAR2)
 VAR1 = SQRT(2.0)

4.3.1.11. Precedence

)

The () operator specifies the order in which an expression should be evaluated. Expression evaluation always begins at the innermost level and processing progresses outward. If no precedence operators are included, then expression evaluation is done according to operator / function precedence. Refer to Table 4-2 for list, they are shown in descending order.

Table 4-2. Precedence of expressions

HIGHEST	CNC NUM CNC TMP CNC STAT CNC POSITIONS CNC MODE	NOT, ABS, COS, SIN TAN, ACOS, ASIN, ATAN INT, FRAC, SQRT ()
MEDIUM	*, /, ^, AND, SHL , SHR	
LOWEST	+, -, OR, XOR	



If all expressions have the same precedence, then evaluation occurs from left to right.

EXAMPLE:

If VAR2 is 2, then

```
VAR1 = 2 * ( VAR2 + 1 )      ;sets VAR1 = 6
VAR1 = (2 * VAR2) + 1      ;sets VAR1 = 5
```

4.3.2. Trigonometric Operators

Each of the trigonometric operators described below only requires one operand. All angular measurements are specified in radians.

4.3.2.1. Sine SIN

This operator produces the sine of the angle specified.

```
VAR1 = SIN( PI/2 )  
VAR1 = SIN( ARRAY1[VAR2] )
```

4.3.2.2. Cosine COS

This operator computes the cosine value of the angle specified as its only operand.

```
EXAMPLE:  VAR1 = COS( PI/3 )  
          VAR1 = COS( VAR2 )
```

4.3.2.3. Tangent TAN

This operator computes the tangent value of the angle specified.

```
EXAMPLE:  VAR1 = TAN( PI/4 )  
          VAR1 = TAN( VAR2/2 )
```

4.3.2.4. Arcsine ASIN

This operator computes the angle whose sine value was specified as the operand. The computed value is in the range of 0 through $\text{PI} * 2$.

```
EXAMPLE:  VAR1 = ASIN( 1.0 )  
          VAR1 = ASIN( VAR2-1 )
```

4.3.2.5. Arccosine ACOS

This operator computes the angle whose cosine value was specified as the operand. The computed value is in the range of 0 through $\text{PI} * 2$.

```
EXAMPLE:  VAR1 = ACOS( 0.5 )  
          VAR1 = ACOS( 1-VAR2 )
```

4.3.2.6. Arctangent ATAN

This operator computes the angle whose tangent value was specified as the operand. The computed value is in the range of 0 through $\text{PI} * 2$.

```
EXAMPLE:  VAR1 = ATAN( 0.25 )
```

4.3.3. Logical Operators

The programming language of the UNIDEX 631/U600 supports logical operations on variables. Although all operands are specified in floating point, the CNC performs the operation in binary. The conversion to and from binary is performed automatically.

4.3.3.1. Negation NOT

This operation produces the one's complement of the binary value of its operand. The result has all bits in the opposite state of the operand.

EXAMPLE: VAR1 = NOT(65535)
 VAR1 = NOT(VAR2)

4.3.3.2. And AND

This operation produces the logical AND of the two operands specified. Therefore, if corresponding bits in both operands are set, the corresponding bit in the result will be also set. Otherwise, that bit will be cleared.

EXAMPLE: VAR1 = VAR1 AND 63
 VAR1 = VAR1 AND VAR2

The fractional portion of VAR1 is discarded before the AND operation.

4.3.3.3. Or OR

This operation produces the logical OR of the two operands specified. Therefore, if either of the operands has a particular bit set, that bit will also be set in the result.

EXAMPLE: VAR1 = VAR1 OR 32
 VAR1 = VAR1 OR VAR2

The fractional portion of VAR1 is discarded before the OR operation.

4.3.3.4. Exclusive Or XOR

This operation produces the exclusive or (logical XOR) of the two operands provided. The result produced is Operand1 with the bits changed which were specified in Operand2.

EXAMPLE: VAR1 = VAR1 XOR 65535
 VAR1 = VAR1 XOR VAR2

The fractional portion of VAR1 is discarded before the XOR operation.

4.3.3.5. Shift Left SHL

This operation performs a logical binary shift (to the left) of Operand1. The number of shifts to be performed is specified using Operand2. The result is equivalent to multiplying Operand1 by 2^{Operand2} . Operand2 must be a positive number.

EXAMPLE: VAR1 = VAR1 SHL 2
 VAR1 = VAR1 SHL VAR2

The fractional portion of VAR1 is discarded before the SHL operation.

4.3.3.6. Shift Right SHR

This operation performs a logical binary shift (to the right) of Operand1. The number of shifts to be performed is specified using Operand2. The result is equivalent to dividing Operand1 by 2^{Operand2} . Operand2 must be a positive number. The fractional part of the operand is discarded before the SHR operation.

4.3.4. Relational Operators

The UNIDEX 631/U600 CNC supports the use of six relational operators used for the comparison of two floating point operands. The operands used by these operators are assumed to be in floating point format, but may be any of the following types:

- User-defined Variable
- System Variable
- Symbolic Constant
- Numeric Literal.

The result produced by each of these operators is Boolean, either TRUE (1) or FALSE (0). They are used extensively by commands such as JUMP, IF and WHILE.

Due to the similarity in syntax, syntax diagrams have been omitted from this discussion. Since none of the commands which use these operators have been discussed yet, programming examples have also been omitted. Please refer to the JUMP, IF and WHILE commands, in Section 4.4, for specific examples of each relational operator.

4.3.4.1. Equal To EQ

If the value of the first number is equal to the value of the second number, the result is TRUE. If the numbers are not equal, the result is FALSE.

4.3.4.2. Not Equal To NE

If the value of the first number is not equal to the value of the second number the result is TRUE. If the numbers are equal the result is FALSE.

4.3.4.3. Greater Than **GT**

If the value of the first number is greater than the value of the second number the result is TRUE. If the value of the first number is less than or equal to the value of the second number, the result is FALSE.

4.3.4.4. Greater Than or Equal To **GE**

If the value of the first number is greater than or equal to the value of the second number, the result is TRUE. If the value of the first number is less than the value of the second number, the result is FALSE.

4.3.4.5. Less Than **LT**

If the value of the first number is less than the value of the second number, the result is TRUE. If the value of the first number is not less than the value of the second number, the result is FALSE.

4.3.4.6. Less Than or Equal To **LE**

If the value of the first number is less than or equal to the value of the second number, the result is TRUE. If the value of the first number greater than the value of the second number, the result is FALSE.

4.4. Commands which Affect Program Flow

As mentioned above, some of the extended commands supported by the UNIDEX 631/U600 CNC provide the ability to define variables, while others permit operations on those variables. The commands described in this section use the variables described above to modify the flow of parts program execution. When analyzing the examples found within this section, please refer to the descriptions of the variables found in the previous sections.

Throughout the discussion below, the terms expression and conditional expression will be found. The term expression refers to a series of mathematical computations which yields a single numerical result. Therefore, expressions can be as simple as the evaluation of a variable or as complex as the solution to an equation. Any of the mathematical operators described in Section 4.3 may be used to form an expression.

The term conditional expression refers to an expression that evaluates to one of two values, TRUE (1) and FALSE (0). This is most often the result of the comparison of two expressions. This comparison may be done using any of the relational operators described in Section 4.3.4 .

4.4.1. Jump to a User Defined Entry Block

JUMP

This command alters the flow of program execution. Instead of proceeding with the next sequential program block, execution immediately proceeds to the entry point specified by the parameter. Program flow may also be changed pending the outcome of a relational operation (see examples on the following page).

SYNTAX: (JUMP, <entry point>) or (JUMP, <entry point> <Condition>)

EXAMPLE:

(DVAR, VAR1)	;Define a variable called variable ONE
(JUMP VALUE0 VAR1 LE 0)	;Jump to the entry point VALUE0 if the value ;of VAR1 is less than or equal to 0
(JUMP VALUE4 VAR1 GT 3)	;Jump to the entry point VALUE4 if the value ;of VAR1 is greater than 3
(JUMP VALUE1 VAR1 EQ 1)	;Jump to the entry point VALUE1 if VAR1 ;contains the value of 1
(JUMP VALUE3 VAR1 NE 2)	;Jump to the entry point VALUE0 if VAR1 ;contains the value of 0
<program block >	;VAR1 must be equal to two
(JUMP, EXIT)	;Jump to the exit point
(DENT,VALUE0)	
<program block >	;VAR1 must be less than or equal to zero
(JUMP, EXIT)	;Jump to the exit point
(DENT,VALUE1)	
<program block >	;VAR1 is equal to one
(JUMP, EXIT)	;Jump to the exit point
(DENT,VALUE3)	
<program block >	;VAR1 is equal to three
(JUMP, EXIT)	;Jump to the exit point
(DENT,VALUE4)	
<program block >	;VAR1 is greater than three
	;Continue to the exit point
(DENT,EXIT)	



The JUMP command should occupy its own block within a program.

The entry points specified as the destination for the JUMP must be defined using the DENT extended command.

4.4.2. Conditional Statement

IF-THEN-ELSE-ENDIF

This construct permits you to execute a group of program blocks only if a specified condition is true. It also provides the capability of executing an alternative group of program blocks if the condition is determined to be false.

As can be seen from the syntax diagram below, the program block containing the IF command must also contain a conditional expression and the keyword THEN. Following this program block are the blocks which are to be executed when the conditional expression evaluates to TRUE, followed by a program block containing ENDIF as the terminator for the IF block. If an action is to be performed when the expression evaluates to FALSE, the group of blocks associated with a TRUE condition is terminated by the ELSE reserved word. Also in this case, the ENDIF reserved word is used to terminate the entire construct.

SYNTAX: *IF <conditional expression> THEN*
 <program block>
 ...
 <program block>
ENDIF
 or
IF <conditional expression> THEN
 <program block>
 ...
 <program block>
ELSE
 <program block>
 ...
 <program block>
ENDIF

EXAMPLE:

(DVAR,VAR1)	;Define a variable named VAR1
IF CNCNUM EQ 1 THEN	;If CNC #1 is executing this program,
VAR1 = 1	;set VAR1 to the value of 1.
ELSE	;Otherwise,
VAR1 = 2	;set VAR1 to the value of 2.
ENDIF	
IF VAR1 NE 2 THEN	;If VAR1 is not equal to 2,
IF GLOBAL00 LT 20 THEN	;and GLOBAL00 is less than 20,
G62 20	;set the profile time to 20.
ENDIF	;Don't do Anything if GLOBAL00 is greater
	;than or equal to 20
ELSE	;If VAR1 is not equal to 2,
G62 10	;set the profile time to 10.
ENDIF	



As demonstrated by the example, IF statements may be nested within one another. The maximum nesting depth is 20 levels.

4.4.3. Repeat Loop

RPT/ENDRPT

It is desirable, in some applications, to have a group of program blocks executed a fixed number of times, unconditionally. A repeat loop is a simple way of performing such a task.

The RPT command designates that the group of program blocks is to be executed multiple times. The number of times to execute this block is specified as a parameter to this command.

SYNTAX: *RPT, <Count>*
 <program block>
 ...
 <program block>
 ENDRPT

EXAMPLE:

(DVAR,VAR1)	;Define a variable named VAR1
RPT,10	;The following program block will be repeated 10 times
G1 G9 X0.1 F100.	
ENDRPT	;End of the repeat loop
VAR1 = 5	;Repeat counts may also be specified in variables
RPT,VAR1	;Enclosed blocks will be executed 5 times
RPT,2	;This inner loop will be executed twice each time the outer loop is executed
G1 G9 X-0.05 F100.	
ENDRPT	;End of the inner repeat loop
ENDRPT	;End of the outer repeat loop

As shown in the example, repeat loops may be nested within each other. The maximum nesting depth is also 20 levels.



4.4.4. Conditional Branch On Error Conditions **ONERRGOTO**

The ONERRGOTO command is used to redirect program flow if a certain condition occurs. It operates much like an interrupt, if the condition occurs anytime during program operation, execution immediately jumps to the label specified. However, there is no mechanism to return to the point where the interrupt occurred. The ONERRGOTO operates like a “JUMP”, not a subroutine.

The user can specify multiple ONERRGOTOs in a program. The priority at which ONERRGOTO will be processed depends on the priority level assigned to the ONERRGOTO through the “P” parameter. The priority assignments are 0 through 100, 0 is a special flag that disables the interrupt (it will not trigger an interrupt) and the highest priority assignment is 100. If one ONERRGOTO is being processed and a second ONERRGOTO with a higher priority becomes true, it will interrupt the current execution. If no priority level is assigned to “P”, the priority default is 50.

SYNTAX: *(ONERRGOTO <label[[V# L#] / G#] [P#] / CLEAR [V# / G#] / ENABLE [V# /G#][P#]>*

EXAMPLES:

To assign an onerrgoto condition:

```
(ONERRGOTO handler)           ;on any CNC or axis fault jump to label handler
(ONERRGOTO handler V432 L0)   ;if virtual bit 432 goes low jump to label handler
(ONERRGOTO handler G50)       ;jump to label handler when variable GLOBAL50 is non-zero
```



Any of these above examples can have a priority assigned to them. For example, *(ONERRGOTO handler V432 L0 P100)* ;same as above with highest priority).

To clear an onerrgoto definition (remove the onerrgoto definition):

```
(ONERRGOTO CLEAR)           ;clear monitoring of CNC and axis faults
(ONERRGOTO CLEAR V432)     ;clear monitoring of virtual bit 432
(ONERRGOTO CLEAR G50)      ;clear monitoring of GLOBAL50 variable
```

To prioritize an existing onerrgoto definition (to the default of 50):

```
(ONERRGOTO ENABLE)         ;enable monitoring of CNC and axis faults
(ONERRGOTO ENABLE V432)   ;enable monitoring of virtual bit 432
(ONERRGOTO ENABLE G50)    ;enable monitoring of variable GLOBAL50
```



Any of these above examples can have a non-default priority assigned to them. For example, *(ONERRGOTO ENABLE V432 P100)* ;same as above with highest priority or temporarily disable monitoring of the condition without removing the onerrgoto definition (so it may be later re-enabled), by setting the priority to 0, then resetting the priority to its original desired value.

The syntax of the ONERRGOTO depends on the condition the user wants the jump to occur. The user can specify an interrupt on a fault, an IO bit value change, and a global variable change. The first parameter must be either, "ENABLE", "CLEAR", or a program label. If it is a program label, the statement defines and activates an ONERRGOTO condition. If it is "CLEAR", it removes the ONERRGOTO monitoring state. If it is "ENABLE", it defines the priority of an existing (active) ONERRGOTO condition. The "#" in the syntax indicates a number must be supplied there. The "P" parameter is optional and can be added to any one of the syntax selections. The "V", "L", "P", and "G" parameters are optional and can appear in any order after the *keyword*.

EXAMPLE:

```

;User interrupt (ONERRGOTO) example program (ERR_GO.PGM)
;Note: this program will run by itself as shown
(DVAR, INT_BIT)
(ONERRGOTO ERR_LBL V0 L1) ;goto line labeled err_lbl when input 0 goes LOW !
;do whatever in your program, such as :
(DENT, LP)
(DISPLAY "Doing something here!")
(JUMP, LP)
M02 ;program end
(DENT ERR_LBL)
(DISPLAY "Entering interrupt handler now
!")
;do what you want here !, then
;Note: M1963 must be defined in your mcode1.ini file as follows :
;"M1963 BI VIRTUAL 0", which defines M1963 to read input bit 0
;get interrupt I/O bit state
M1963 $INT_BIT
WHILE INT_BIT EQ 1 DO
M1963 $INT_BIT
G4 F.5 ;switch bounce delay
ENDWHILE
(ONERRGOTO CLEAR V0) ;clear interrupt condition (REQUIRED !)
INT_BIT = CNCFAULT ;read the cnc fault
(ONERRGOTO ERR_LBL V0 L1) ;enable interrupt to occur again !
(DISPLAY "Leaving interrupt handler !")
(JUMP, LP) ;jump to somewhere in your program

```

To respond to axis or programming errors, the FAULTMASK and INTERRUPT masks must be set to program the desired fault condition as well as the CNC fault. See the *U631/U600 User's Manual* for additional information on setting FAULT and INTERRUPT masks.



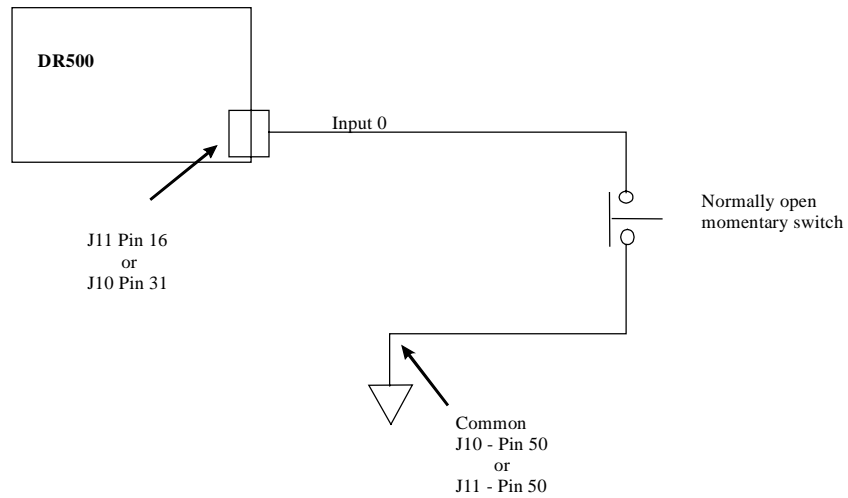


Figure 4-1. U600 User Interrupt

4.4.5. Conditional Looping

It is desirable, in many applications, to have a group of program blocks repeatedly executed until a specific condition becomes true. The WHILE-DO-ENDWHILE construct, provided by the UNIDEX 631/U600 CNC programming language, permits such a function to be implemented easily.

As can be seen from the syntax diagram below, the program block which contains the WHILE command must also specify a conditional expression to be evaluated and the keyword DO. The entire construct must be terminated using the ENDWHILE keyword.

During execution, the CNC Processor evaluates the conditional expression specified. If TRUE, it executes the group of program blocks immediately following this block. When the ENDWHILE keyword is encountered, program flow returns and evaluates the conditional expression again. This will be performed repeatedly until the condition is evaluated as FALSE. Note that if the condition is FALSE on the first evaluation, the body inside the WHILE will never be executed. At this time, program flow proceeds to the statement immediately following the ENDWHILE keyword.

SYNTAX: *WHILE* <conditional expression> *DO*
 <program block>
 ...
 <program block>
 ENDWHILE

EXAMPLE:

(DVAR,OUTER,INNER)	;Define variables OUTER and INNER
OUTER = 0	;Initialize outer loop counter to zero
WHILE OUTER LT 10 DO	;Execute this loop until the variable OUTER
	;is greater than or equal to 10
OUTER = OUTER + 1	;Increment the loop counter
ENDWHILE	;End of while construct
WHILE OUTER GE 1 DO	;Execute this loop until the variable OUTER
	;is less than or equal to 0
INNER = 0	;Initialize inner loop counter
WHILE INNER LE 5 DO	;While loops can be nested inside of one
	;another
INNER = INNER + 1	
IF INNER EQ 5 THEN	;Even conditional statements can be in while
	;loops
OUTER = OUTER -1	
ENDIF	
ENDWHILE	
ENDWHILE	

As shown in the example above, WHILE loops can be nested within one another. The maximum nesting depth is 20 levels.



4.4.6. Call Subroutine/Call Library Subroutine**CLS/CLLS**

This command calls a subroutine. When the specified subroutine has completed its task, program execution continues with the next program block.

Information, such as the location from which the subroutine was called, is stored on the user's stack. If the specified subroutine is called as a library subroutine, information reflecting the current operational mode will also be stored there. Operating mode information will be restored when returning from a library subroutine call.

SYNTAX: (*CLS*, <subroutine name>) or (*CLLS*, <subroutine name>)

EXAMPLE:

(CLS ,MOVEHOME)	;Call the subroutine "MOVEHOME"
-----------------	---------------------------------



A subroutine must be defined, using the DFS commands, prior to being called.

Subroutines are permitted to call other subroutines.

4.4.7. Execute OS/2 or DOS Program**EXECUTE**

This command allows the operator to execute any .EXE file, .COM file, .CMD or .BAT file as a CNC command. Any set of parameters (up to 100 characters) may be passed to the executable file and variable values may be included in the parameters. By providing a “null” executable name (see examples below) the operator can specify that an OS/2 window be displayed, allowing entry of commands. The CNC program will wait until the executable file is completed or for a “null” executable file before continuing to the next step in the CNC program.

Also, the operator can run .BAT files with the special syntax and receive the return code from the executable file and place it into a CNC variable (see example below).

SYNTAX: (*EXECUTE FileName.EXE*) or (*EXECUTE FileName.EXE [=VAR]*)

EXAMPLE:

(EXECUTE “ “)	;Opens up OS/2 window.
(EXECUTE “vdiff.exe a.dat b.dat” = ERR1)	;Runs the vdiff program, return code ;from vdiff is put in ERR1 variable.
(EXECUTE “Cleanup.exe “ \$VAR1 “/a”	;Runs Cleanup.exe with \$VAR1 and ;/a as parameters. For example, if ;VAR1 was currenty 10, the call ;would be: Cleanup.exe 10 /a.
(EXECUTE “Command.com /C typeit.bat”)	;Runs the batch file typeit.bat.

The executable file must exist somewhere within the OS/2 path, not necessarily in the U31 directory.



4.5. Custom Display Window Commands

The UNIDEX 631/U600 CNC provides the ability for the parts program to interface with the operator via a message window referred to as the Custom Display Window (CDW). The commands found in this section permit the operator to open and close the window, as well as display information within the window and receive input from the operator.

4.5.1. Open Custom Display Window

OPENCDW

This command causes the Custom Display Window to appear on the CNC Run Mode Screen. The size and initial orientation of the window is pre-defined, however, the operator is free to move it as desired.

Initially, the list box found in this window is empty. Messages may be displayed using the DISPLAY command discussed below. The window will remain open until the CLOSECDW command is executed.

SYNTAX: (*OPENCDW*)

EXAMPLE:

(OPENCDW)	;Opens the Custom Display Window
-----------	----------------------------------



The OPENCDW command is superfluous because a DISPLAY command with no preceding OPENCDW will automatically open the display window.

Please refer to the comprehensive example following the DISPLAY command description.

This command must occupy its own program block.

Issuing an OPENCDW command while the Custom Display Window is open has no effect.

4.5.2. Close Custom Display Window**CLOSECDW**

This command causes the Custom Display Window, created with the OPENCDW command, to disappear from the CNC Run Mode Screen. All data displayed within the window, using the DISPLAY command, is lost. See RECORDON statement to preserve this data in a disk file.

SYNTAX: (*CLOSECDW*)

EXAMPLE:

(CLOSECDW)	;Close the Custom Display Window
------------	----------------------------------

Please refer to the comprehensive example following the DISPLAY command description.

This command must occupy its own program block.

A CLOSECDW command is ignored if the Custom Display Window is not open.

The Custom Display Window closes automatically upon program completion or termination.

**4.5.3. Placing Items in the Window****DISPLAY**

This command prints a message into the Custom Display Window (CDW). The message occupies exactly one line of text in the CDW window. There are many options based on the parameters provided (see examples that follow). The most important option is whether the program should continue on after printing the message, or wait for the user to respond.

If the display window already had messages on it, the new message is placed on a new line below the previous message. A scroll bar is available that permits the user to browse through previous messages. If more than 20 messages are delivered to the same display window, the program discards the oldest message and the text “...” is placed on the last line indicating that a message has been discarded.

The line of text and waiting behavior is determined by the parameters in the DISPLAY statement. The statement can wait for a user response of a pushbutton, wait for the user to enter a value, or not wait at all. Any number of parameters may be provided (up to the length of a line) in any order. The parameter syntax is best understood by studying the examples in DISPLAY.PGM, it is not explained here.

Each text string is restricted to 39 characters. However, longer strings can be displayed by concatenating multiple strings.

SYNTAX: (*DISPLAY specifier*)

Where any number of *pspecifiers* can be in the parameter list. The syntax of a *pspecifier* is described in the example below.

EXAMPLES:

(DISPLAY "Performing program startup")	;This statement displays the text without waiting for ;or accepting any user response. The display window ;is opened, if not already open
(Display "You can print very very very extremely" "long strings like this one")	;Shows how to print large strings.
(CLOSECDW)	;This code erases all previous messages in the list and ;makes the given message the first one on the list
(OPENDCW)	
(DISPLAY "Brand new message")	
(DVAR WIDTH HEIGHT)	;This code prints a string inserting the integer value ;where shown. The CNC programmer has no choice ;over the format of the value shown.
WIDTH = 1	
HEIGHT = 2	
(DISPLAY "The square is: " WIDTH "millimeters. by " HEIGHT "millimeters.")	
(DVAR RESPONSE)	;This code pauses the program after delivering the ;message offering the user three buttons to push. ;"OK", "continue", or "no".
(DISPLAY "The drive has overheated" =RESPONSE)	
(DISPLAY "The drive has overheated" =RESPONSE /BUTTON2)	;this is the same as the code above, except the ;"continue" button is not shown.
(DISPLAY "The drive has overheated" =RESPONSE /BUTTON1)	;same comment, except only the "OK" button is ;shown.



1, 2, or 3 buttons can be specified. When the operator responds "OK", a zero "0" is placed in the specified variable ("RESPONSE" in the example above). "NO" produces a 1, and "Continue" produces a 2.

The following example (program fragment) shows how messages can be stacked and complex dialogues maintained with the user. The “/INTEGERENTRY” keyword forces the operator to enter an integer (no floating point). If the keyword “/VALUEENTRY” were used a floating point could be entered. The “=” specifies what variable the value entered by the operator is placed.

```
(DVAR TEMP2, PGAIN, DANGER_LIMIT)
(CLOSECDW)
(DISPLAY “PGAIN is invalid”)
(DENT GETPGAIN)
(DISPLAY “Enter a new proportional gain.” =PGAIN /INTEGERENTRY);
IF PGAIN LT 0 THEN
  (DISPLAY “That number:” PGAIN “is too low.”)
  (JUMP GETPGAIN)
ELSE IF PGAIN GT 10000000 THEN
  (DISPLAY “That number:” PGAIN “is too high.”)
  (JUMP GETPGAIN)
ELSE IF PGAIN GT DANGER_LIMIT THEN
  (DISPLAY “That number: “PGAIN “seems large, do you want “ “to go with it anyway” = TEMP2)
  IF TEMP2 EQ 1 THEN ; user said no
    (JUMP GETPGAIN)
  ENDIF
ENDIF
(CLOSECDW)
```

This command must occupy its own program block.

Individual quoted strings may not exceed 39 characters.

If a “/” option (i.e., /BUTTON1) is used, then a “=var” must also exist in the statement.



4.5.4. Activate CDW Log File**RECORDON**

This command enables logging of all messages displayed within the Custom Display Window (CDW) to the specified file. All subsequent DISPLAY commands will cause their output to be displayed within the CDW, as well as written to the CDW Log file. This mode of operation is terminated using the RECORDOFF command.

The only parameter to this command is the name of the file in which the information is to be stored. This parameter may be any valid OS/2 file name (100 chars max.) and may be specified using either absolute or relative path names. If no path name is specified, the current program directory is assumed (\U31\PROGRAMS by default). If the file specified already exists, the old file is discarded.

SYNTAX: (*RECORDON FileName.Ext*)

EXAMPLE:

(RECORDON DISPLAY.DAT)	;Causes all data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the \U31\PROGRAMS directory.
(RECORDON \U31\PART01\DISPLAY.DAT)	;Causes all data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the \U31\PART01 directory.
(RECORDON PART01\DISPLAY.DAT)	;Causes all data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the PART01 sub-directory of the ;directory from which the CNC was invoked.

For a more detailed example, refer to the RECORDOFF command.



If this command is encountered when a log file is already open, the command is ignored.

4.5.5. Deactivate CDW Log File

RECORDOFF

This command disables logging messages displayed within the Custom Display Window (CDW) to the log file opened with the RECORDON command. Subsequent messages displayed within the CDW will no longer be placed into the log file.

SYNTAX: (*RECORDOFF*)

EXAMPLE:

(OPENCDW)	;Activate the Custom Display ;Window
(RECORDON DISPLAY.DAT)	;Causes all data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the \U31\PROGRAMS directory
(DISPLAY "This is a Test")	;This message will be logged to the file
(RECORDOFF)	;Close the log file
(RECORDON \U31\PART01\DISPLAY.DAT)	;Causes all new data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the \U31\PART01 directory
(DISPLAY "Some Text")	;This message will be logged to the file
(RECORDOFF)	;Close the log file
(RECORDON PART01\DISPLAY.DAT)	;Causes all data displayed in CDW to be logged to file ;DISPLAY.DAT, found in the PART01 sub-directory of the ;directory from which the CNC was invoked
(DISPLAY "Some Text")	;This message will be logged to the ;file
(RECORDOFF)	;Close the log file
(CLOSECDW)	;De-activate Custom Display Window

This command has no effect if the Custom Display Window log file is currently not open.

If program execution ends without executing this command or is terminated by the operator, the log file closes automatically.



4.6. Synchronous Motion Commands

In a synchronous motion the CNC program suspends execution until the motion indicated by the command is complete.

4.6.1. ENDM Command

ENDM

The *ENDM* command ends motion on a single axis. If there is currently no motion it has no effect. It is the functional opposite of the *STRM* command. The *ENDM* command exhibits the same decel behavior as a *G1* or *G0*. The *ENDM* command is synchronous, meaning the CNC will wait until the motion is actually stopped before proceeding to the next CNC program step.

SYNTAX: *ENDM axispec*

EXAMPLE:

(ENDM Y)	;End the master axis motion
----------	-----------------------------

4.6.2. Free

FREE

The FREE statement will deallocate the memory that was previously allocated to a cam table. The memory is on the U600/U631 motion card.

SYNTAX: (*FREE, table_number*)

EXAMPLE:

(FREE, 1)

4.6.3. Handwheel Command

HAND

The HANDwheel command permits the user to manually position an axis with a handwheel having a standard quadrature encoder output (or with any other device having the same). The handwheel is connected to the U600/U631 through a spare encoder channel input so X4 multiplication will be done on the handwheel quadrature signal producing four times the number of counts per revolution specified by the manufacturer. Some handwheels actually produce four counts per step of the handwheel. Large values for the distance parameter will produce jerky motion possibly causing one of the axis traps to disable the axis. Disabling the VFF and AFGAIN parameters will provide smoother handwheel operation (see the SETPARM command).

The encoder channel parameter may specify any of the 16 possible encoder channel inputs 1 through 16 (channels 5 through 16 are located on the encoder expansion cards 1 through 3 respectively). It may also specify the commanded positions of axes 1 through 16 by designating an encoder channel of 17 through 32 respectively. Specify encoder channel zero (0) or distance of zero (0) to disable the handwheel command for a particular axis.

Multiple hand commands may be executed sequentially to permit simultaneous multi axis positioning.

SYNTAX: (HAND, encoder_ch, axis, dist_per_encoder_count)

Where

encoder_ch - specifies the encoder channel the handwheel is connected to(0-32) axis is the axis to be positioned by the handwheel.
 dist_per_encoder_count - is the distance the axis will move for each count of the handwheel.

(DVAR, RESPONSE)	
(Hand, 3, X, .01)	;Position X axis via handwheel on encoder channel 3
(Hand, 4, Y, .1)	;Position Y axis via handwheel on encoder channel 4
(Display "Press OK when finished positioning "= RESPONSE)	
(Hand, 0, Y, .1)	;Disable Y positioning
(Hand, 0, X, .01)	;Disable X positioning

4.6.4. Home

HOME/REF

The HOME and REF commands accept parameters that specify which axes are to be affected. Using an axis name in the parameter list designates that this axis is to move to its hardware reference point. This command causes the specified axes to initiate a homing sequence. These commands provide the ability to locate an axis or set of axes at a known point in space. The exact location of this point is determined by the "home type" machine parameter that typically uses a reference switch and the marker pulse from the optical encoder (providing a highly repeatable reference).

Once this command has been issued, parts program execution is suspended until all axes specified have reached their respective hardware home positions. As each axis reaches this position, all position registers associated with that axis are set to zero.

SYNTAX: (HOME, <axis name>, <axis name> ...)

EXAMPLE:

(HOME,X,Y)	;The X and Y axes are sent home simultaneously
(REF Axis_01, Axis_02)	; Initiate a homing sequence simultaneously on Axis_01 and Axis_02

The type of homing sequence performed is specified using the Home Type entry field of the Machine Parameter Menu.

The HOME and REF commands may be used interchangeably.

The user cannot home a virtual axis, the axis will move indefinitely.



4.6.5. Slew Command**SLEW**

The SLEW command allows the user to position the axis manually through the use of a RS-232 serial trackball or mouse (U631) or an analog joystick (U600). On the U631 the port number represents the serial port on the axis processor board that connects to the trackball and/or mouse (0-3).



The mouse and trackball must be connected to the desired serial port on the U631 axis processor board when its firmware is loaded (on power up).

On the U600 the port refers to the connection of the joystick. Port 0 refers to the joystick port on the U600 board, 1 to 3 refers to the joystick port on the optional encoder expansion boards 1 to 3. In all cases the joystick connects to the joystick connector on the DR500, BB500 or BB501 that interfaces the user to the U600 board.

The *speed* is the maximum velocity that may be commanded by the input device. The *x* parameter is the axis designator commanded by the movement of the X axis of the slew device. The *y* parameter optionally specifies the axis designator commanded by the movement of the Y axis of the slew device. To disable the slew mode press the fire button on the joystick or press the left button on the mouse or trackball.

SYNTAX: (*SLEW, port, speed x[, y]*)

Where

- port* - is either the serial port (U631) or the joystick port (U600).
- speed* - is the maximum speed in user units.
- x* - is the axis to be commanded by the x axis movement of the slew device.
- y* - is the optional second axis to be moved by the yaxis motion of the slew device.

EXAMPLE:

(SLEW, 0, 1000., x, y)	;Activates the trackball through port 0
------------------------	---



Speed is specified in user units.

4.7. User Stack Operations

As mentioned above, your stack maintains all program information such as the program line to return to when a subroutine is called, and the current operational mode when calling a library subroutine. In these cases, all stack operations are performed without user intervention.

The *UNIDEX 631/U600* CNC programming language also permits you to place information on and retrieve information from this stack. The commands described in this section are used to perform these functions. However, caution should be exercised when manipulating data on the stack.

In general, a stack is a data structure in which the value that was stored most recently must be removed before the older data may be removed. In order to help understand the concept, the reader may think of each data item as a plate and your stack as a stack of these plates. When one plate is added to the stack, it is placed on top of the others. In order to retrieve the plate which is third from the top, the top two plates must be removed first.

4.7.1. Putting Data Onto the User's Stack

PUSH

This command places data onto your stack. The value of each parameter is sequentially placed on top of the stack. The stack pointer is automatically incremented as each item is stored. The information stored with this command may be retrieved using the POP command described below.

SYNTAX: (*PUSH <data>*)

EXAMPLE:

(PUSH VAR1)	;Place the value of VAR1 onto your stack
-------------	--

Please refer to the comprehensive example following the POP command on following page.

All information placed on the stack within a subroutine must be removed from the stack prior to subroutine completion. Otherwise, when attempting to return from that subroutine, an incorrect return address will be removed.



4.7.2. Removing Data From the User’s Stack

POP

This command removes data from your stack. The value(s) are removed from the top of the stack and placed into the variables specified in sequential order. The stack pointer is automatically decremented as each item is removed. This information was previously stored using the PUSH command described above.

SYNTAX: (POP <Variable>)

EXAMPLE:

```
(POP VAR1) ;Remove the value from the top of the stack and place it into VAR1
```

Please refer to Figures 4-2, 4-3, and 4-4 on the following pages for a comprehensive example.

Push/Pop Example

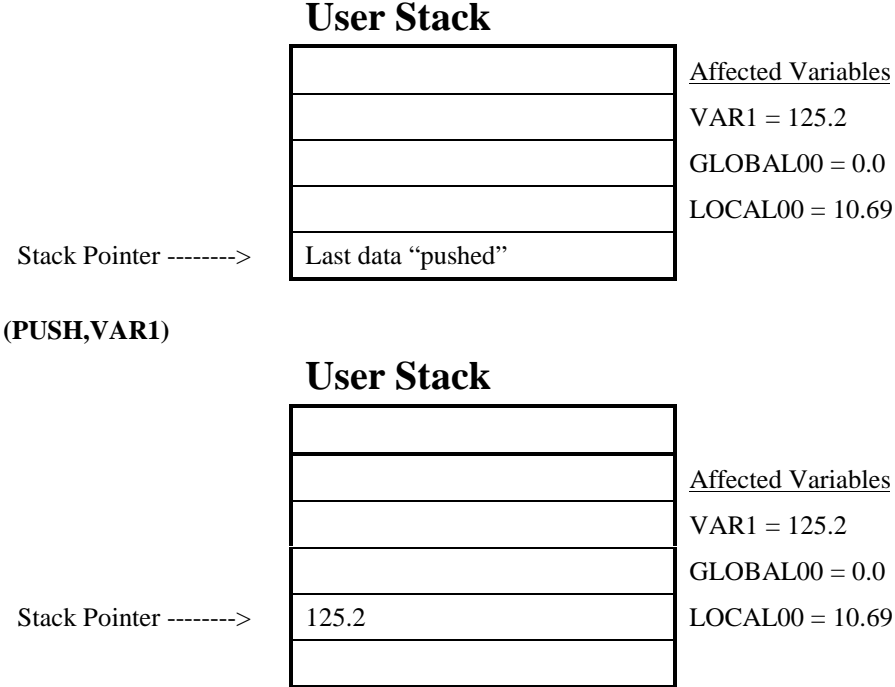
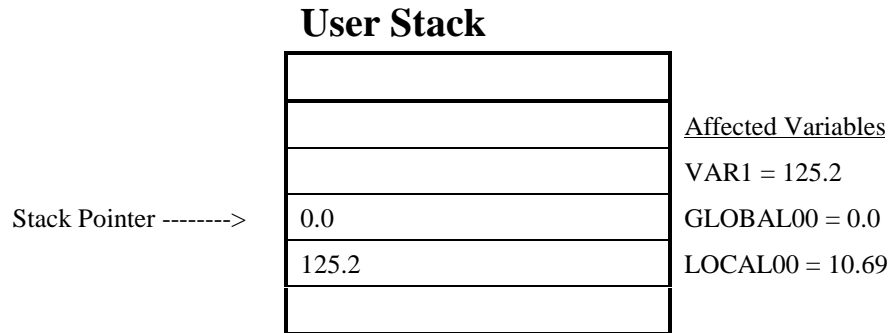
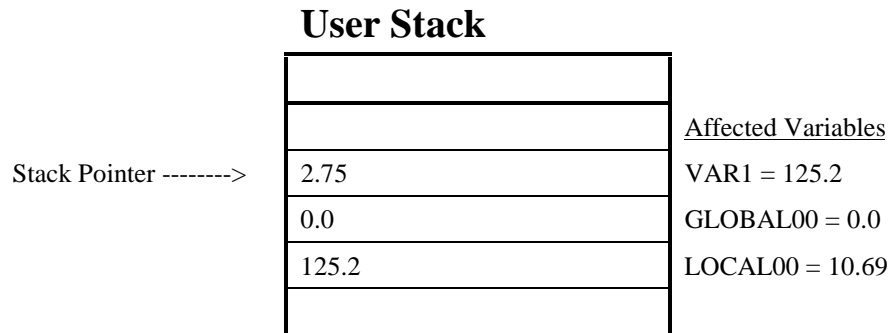


Figure 4-2. Push/Pop Example

(PUSH,GLOBAL00)



(PUSH,2.75)



(POP,VAR1)

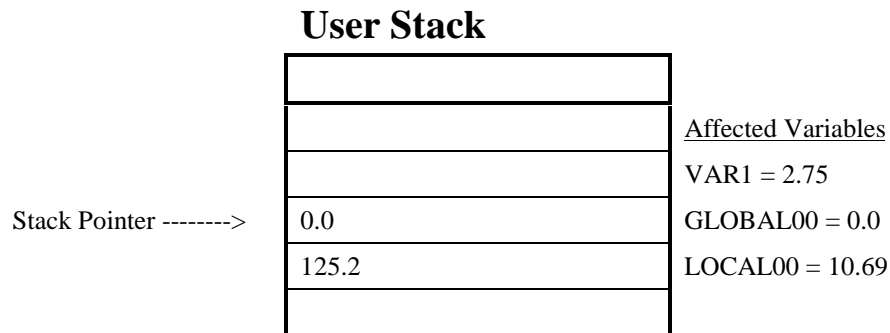
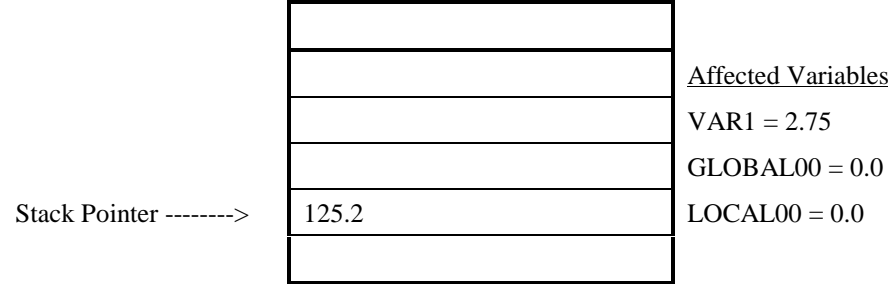


Figure 4-3. Push Global Example

(POP,LOCAL00)

User Stack



(POP,GLOBAL00)

User Stack

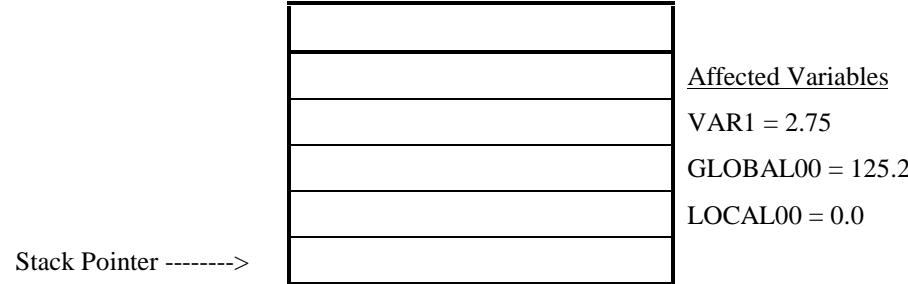


Figure 4-4. Push Local Example

4.8. Miscellaneous Extended Commands

4.8.1. The Autofocus Command

AFCO

The autofocus command allows the user to position an axis in relation to an optimum point determined by feedback from an analog input device. This device produces an analog voltage proportional to the position error. This error is scaled to the maximum speed specified (with 10 volts representing maximum speed) and commands the specified axis to move in the direction relative to the zero point of the autofocus device. The offset allows the user to specify a non-zero voltage point where the optimum position has been reached. The AFCO command may be disabled by specifying a speed of zero.

The deadband is a positive voltage value centered around the offset voltage, where no motion command is generated. For example, if the offset is 1 volt and the deadband is .5 volts no motion will be generated when the autofocus device produces a voltage between .5 volts and 1.5 volts. A voltage of 1.6 volts would generate a velocity command of $.1 / 10 * \text{speed}$.

On the U631 a channel may be specified in the range of 0 to 7 which represents one of the 8 channels on the optional Matrix analog input card. On U600 a channel may be specified in the range of 0 to 3 which represents one of the 4 channels on the U600 board. However, channels 2 and 3 are normally used for the analog joystick inputs and channel 0 is used when the analog MFO input is activated leaving channel 1 never preassigned to another function.

SYNTAX: (*AFCO, channel, axis, speed, deadband, offset*)

Where

- | | |
|-----------------|--|
| <i>channel</i> | - is the analog input channel. |
| <i>axis</i> | - is the axis commanded by the autofocus device. |
| <i>speed</i> | - is the maximum velocity in user units. |
| <i>deadband</i> | - is the voltage range where no correction is generated. |
| <i>offset</i> | - is the voltage from the focus device at the desired position (null point). |

4.8.2. Axis Naming

HARDNAMES/SOFTNAMES

As mentioned, the UNIDEX 631/U600 CNC permits axis designations to be specified in two ways. By default, each axis may be referred to using the name specified in the "Axis Name" entry field in the Machine Parameter Menu (refer to the *UNIDEX 631/U600 User's Manual*). When referring to an axis using this name, the user is said to be applying the "soft name" for the axis. The term implies that the name may easily be changed.

Alternatively, the UNIDEX 631/U600 CNC has a pre-defined set of axis names which may be used. These names are called "hard names" due to the fact that they may not be changed by you. The set of "hard names" used by the UNIDEX 631/U600 is defined as follows.



Table 4-3. Hard Axis Names

HardName	Axis Number	HardName	Axis Number
X,Y,Z	1-3	x,y,z	10-12
U,V,W	4-6	u,v,w	13-15
A,B,C	7-9	a	16

The HARDNAMES command specifies that these pre-defined names are to be used, as opposed to the names specified in the Machine Parameter Menu. The SOFTNAMES command has the opposite effect.

These commands are modal. This mode is retained until changed, even between invocations of the CNC.

SYNTAX: (*HARDNAMES*)
 or
 (*SOFTNAMES*)

EXAMPLE:

(HARDNAMES)	;Permit specification of HARD axis names
G92 X0 Y0 Z0	;Use the pre-defined axis names to clear the preset registers
(SOFTNAMES)	;Permit specification of SOFT axis names
G92 Axis_01 0 Axis_02 0 Axis_03 0	;Use softnames to clear the preset registers



These commands must occupy a program block of their own.

As mentioned, this command is modal. The specified operational mode is retained across CNC invocations.

4.8.3. Joining Parts Programs**INCLUDE**

This command provides the parts programmer the ability to merge several files together to execute as one, similar to subroutines. The combined file is treated as though all included program blocks were found in the main program file.

The implications of this are that all definitions in the main file are also accessible to the included file. This includes variables, subroutines and entry points. However, it also implies that there may be no definitions with the same name found in the included files.

The filename specified may be any valid OS/2 file name and may be specified using either absolute or relative path names. If no path name is specified, the program is assumed to be present in the current program directory (\U31\PROGRAMS by default).

SYNTAX: (*INCLUDE FileName.Ext*)

EXAMPLE:

(INCLUDE CLEANUP.PGM)	;Include the program "Cleanup.pgm"
(INCLUDE \U31\PART01\SETUP.PGM)	;Include the program "Setup.pgm" found in the ;\U31\PART01 directory
(INCLUDE PART01\CUTPART.PGM)	;Include the program "CutPart.pgm" found in the PART01 ;sub-directory. This path is relative to the location from ;which the CNC was invoked.

If the filename begins with a numeral, an absolute file spec (path) must be specified to prevent the filename from being interpreted as a numeric symbol.

No more than 6 include files may be used within a program and cannot be nested more than 4 levels deep.



4.8.4. Data Collection Control

DATA

The UNIDEX 631/U600 CNC is equipped with a data acquisition feature used to record a variety of information relevant to the performance of a particular axis. The type of data recorded is the commanded position, actual position, actual velocity, torque command, etc.

The DATA command controls the operation of this function within the executing parts program. The command parameters allow operations such as entering/exiting the data collection mode and starting/stopping data acquisition to be performed. The collected data is stored in a file which can be displayed using the PLOTDATA diagnostic utility (refer the *UNIDEX 631/U600 User's Manual* for more information).

The first parameter to this command is a keyword that specifies the particular action which is to be performed. All acceptable values for this parameter have been described below.

OPEN

This keyword enters the data collection mode and requires three additional parameters. The first parameter is a list of axes for which data should be collected. These axes should be specified using the naming convention currently active (HARDNAMES/SOFTNAMES).

The second parameter to this command is the rate at which the sampling is to be performed. This rate is specified in milli-seconds (1 msec).

The last parameter specifies the name of the file in which the data acquired is to be saved. This filename specified may be any valid OS/2 file name (80 chars max.) and may use either absolute or relative path names. If no path name is specified, the file is placed into the current program directory (\U31\PROGRAMS by default).

OPEN

When the DATA OPEN is encountered within a parts program, the CNC performs all initialization required for the data collection mode and opens the specified data file. It then displays the Data Acquisition Dialog Box to show this mode is active. As mentioned in the *UNIDEX 631/U600 User's Manual*, this box shows the name of the file to which the data is being logged and the number of acquisitions which have occurred.

If a DATA OPEN command is encountered when the data collection mode is active, any acquisition that is in progress is terminated and the data file closes. Data collection parameters will then be re-initialized and the new data file opened. Subsequent acquisitions will be placed into the new data file.

It should be noted that this command does not actually initiate the acquisition of data. It simply performs all initialization required for operation in the data collection mode. Acquisition is initiated using the START keyword.

The OPEN keyword may have one of three optional keywords following it. If none of these 3 keywords follow it the data output to the file will be in an ASCII text format. The optional keyword producing a data file most similar to this is the BINARY_DATA keyword which produces the same data within the file except the data is in a binary format (as produced by older versions of mainmenu.exe). The second of the optional keyword is NO_AXIS_DATA which produces a data file in an ASCII text format without any axis data. The last of the optional keywords is AXIS_DATA_ONLY which produces an ASCII text file with only axis data.

For example:

```
(DATA OPEN X, Y, 1 data.dat)
```

Sets data collection for the X-Y axis. All system data (Analog inputs, I/O, etc.) and axis data (position, velocity, acceleration, etc.) will be collected at 1msec and written in an ASCII text format to DATA.DAT.

```
(DATA OPEN NO_AXIS_DATA, 1 data.dat)
```

All system data will be collected at 1 msec and written in an ASCII text format to DATA.DAT.

```
(DATA OPEN AXIS_DATA_ONLY, X, Y, 1 data.dat)
```

All X-Y axis data will be collected at 1 msec and written in an ASCII text format to DATA.DAT.

```
(DATA OPEN BINARY_DATA, X, Y, 1 data.dat)
```

All X-Y axis data will be collected at 1 msec and written in a binary format to DATA.DAT. This format is compatible with versions of mainmenu.exe prior to version 4.05.

- START** This keyword initiates the acquisition of data once the system has been placed into the data collection mode (using the DATA OPEN command). Data will then be acquired at the rate specified when this mode was initialized. All acquisitions will be logged to the file specified at that time.
- A DATA START command encountered when the system is not in the data collection mode will be ignored.
- STOP** This keyword terminates any data acquisition currently in progress. This command will be ignored if the system is not in the data collection mode and in the process of acquiring data.
- CLOSE** This keyword exits the data collection mode. The file in which acquired data is being placed is closed, as is the Data Acquisition Dialog Box. The data collection mode must then be reinitialized, using the DATA OPEN command, before more acquisitions can be made.
- Any acquisition in progress at the time this command is executed terminates automatically. A DATA CLOSE command encountered when the system is not in the data collection mode is ignored.

SYNTAX: (DATA OPEN [NO_AXIS_ DATA / Axis_data_Only / Binary_Data] AxisList Rate FileName.Ext)
 or
 (DATA START)
 or
 (DATA STOP)
 or
 (DATA CLOSE)

EXAMPLE:

(DATA OPEN X Y Z 1 DATAFILE.PLT)					
					;Initialize data collection mode.
					;Acquisitions will occur for the X,
					;Y and Z axes at a rate of 0.001
					;seconds (every millisecond). All
					;data acquired will be placed into
					;the file DATAFILE.PLT in the
					;\U31\PROGRAMS directory.
(DATA START)					;Start acquiring data
<program block>					;Any number of program blocks
...					;Typically, these block do motion
<program block>					
(DATA STOP)					;Stop acquiring data
(DATA CLOSE)					;End the data collection mode
(DATA OPEN X	10				;Initialize data collection mode.
\U31\PART01\DATAFILE.PLT)					;Acquisitions occur for the X axis
					;at a rate of 0.010 seconds (every
					;10 msec). All data acquired will
					;be placed into the file
					;DATAFILE.PLT in the
					;\U31\PART01 directory.
(DATA OPEN X	10				;Initialize data collection mode.
PART01\DATAFILE.PLT)					;Acquisitions will occur for the X
					;axis at a rate of 0.010 seconds
					;(every 10 milliseconds). All data
					;acquired will be placed into the
					;file DATAFILE.PLT in PART01
					;sub-directory. This path is
					;relative to the location from
					;which the CNC was invoked.

4.8.5. Reading Axis Parameters**GETPARAM**

This command reads the current value of any axis parameter. Any axis parameter which may be viewed using the STAT command of the ZSID/debug960 utility program may also be accessed using this command. Please refer to the documentation provided with this utility program for a list of all valid axis parameters (refer to Appendix A "Axis Parameters").

The GETPARAM command requires several parameters. The first of these is a list of axes to which this command applies. These axes should be specified using the naming convention currently active (HARDNAMES/SOFTNAMES). This list may contain any number of axes, but all axes mentioned must be associated with this CNC.

The second parameter to this command is the name of the axis parameter whose value is to be retrieved. As mentioned, a list of valid parameter names may be obtained using the ZSID/debug960 utility.

The last parameter to this command is the name of the variable in which the parameter value is to be placed. If multiple axes are specified, the variable specified should be an array element. The value of the first axis specified will be placed into the element specified. Subsequent elements will contain the values retrieved for the other axes.

SYNTAX: (*GETPARAM Axis AxisParameter Variable*)
 or
 (*GETPARAM AxisList AxisParameter StartingArrayElement*)

EXAMPLE:

<code>(GETPARAM X IAVG VAR1)</code>	<code>;Read the current value of the axis ;parameter IAVG for the X axis. ;Place the value of this parameter into ;the variable VAR1.</code>
<code>(GETPARAM X Y Z DRIVE TSTARRAY[0])</code>	<code>;Read the value of the DRIVE axis ;parameter for the X, Y and Z axes. ;The values will be placed into ;TSTARRAY[0], TSTARRAY[1] ;and TSTARRAY[2] respectively.</code>

4.8.6. Initialize Touch Probe

G51

The UNIDEX 631/U600 CNC provides support for digital touch probe measuring. It is designed to permit you to determine the location of the part in space.

The “probe” command initializes the touch probe the G51 command activates probe monitoring. When probe input is detected, the CNC aborts the move in progress and returns the current position of each axis. This enables you to start the part moving toward the probe and have the part stop moving when it reaches the probe. The program may then use the position information returned to determine the physical location of the part in space.

Once a probe cycle is initiated, the CNC actively monitors the appropriate input channel until the probe input is detected. When a probe touch occurs the cycle is complete. To initiate a probe measuring cycle again, execute another G51 command.

The parameters for the probe command include the virtual IO number, active level and variable to store the axes positions. The first parameter, channel number, refers to the virtual input channel through which the probe input will be monitored. The second parameter, active level, permits the user to specify the polarity of the probe in use. The final parameter specifies the first location of an array into which the position information is stored.

Refer to the extended command DVAR for more information on arrays.

SYNTAX: (*PROBE 25 0 POS[0]*) ;*Initialize probe parameters*

G51 ;*Monitor touch probe input*

The G51 command can be used after a probe G51 cycle has been executed. This will enable the touch probe again without repeating the channel level and variable parameters. A G51 has no arguments.

EXAMPLE:

(DVAR,POS<16>)	;Define variable to hold positions
(Probe 10 0 POS[0])	;Initialize touch probe input on virtual input bit 10. The probe being used is active low. Positional information will be placed into the POS array.
G51	;Start motion toward probe
G1 X5.0 F30	;Enable touch probe (same probe parameters as last command)

The modes display will show “G51” until the probe input has been detected.



4.8.7. Modifying Axis Parameters**SETPARM**

This command changes the current value of any axis parameter. Any axis parameter which may be modified using the ZSID/debug960 utility program may also be modified using this command. Please refer to the documentation provided with this utility program for a list of all valid axis parameters (see Appendix A “Axis Parameters”).

The SETPARM command requires a similar parameter list to the GETPARM command. The first of these is a list of axes to which this command applies. These axes should be specified using the naming convention currently active (HARDNAMES/SOFTNAMES). This list may contain any number of axes, but all axes mentioned must be associated with this CNC.

The second parameter to this command is the name of the axis parameter whose value is to be modified. As mentioned, a list of valid parameter names (listed in Appendix A) may be obtained using the ZSID/debug960 utility.

The last parameter to this command is the value to which this parameter is to be set. This value is specified in floating point format and may be specified using a numeric literal, a variable or a simple expression. This value will be applied to all axes specified.

SYNTAX: (*SETPARM AxisList AxisParameter Value*)

EXAMPLE:

(SETPARM X IMAX ;Set the value of the IMAX parameter for the X axis equal to 10000 10000)
(SETPARM X Y Z DRIVE ;Sets the DRIVE parameter for the X, Y and Z axes to the value VAR1) specified ;by VAR1

4.8.8. Monitor Axis Speed**MONSPD**

The MONSPD statement monitors the speed of a specified axis (when it is not accelerating or decelerating). It generates a CNC fault and sets bit 1 (decimal value 2) of the CNCSTAT3 variable if the speed falls outside the designated range. The minimum and maximum limits designated are in user units per second. The next statement in a CNC program executes immediately after this statement.

SYNTAX: (*MONSPD, axis, min_limit, max_limit*)

EXAMPLE:

(MONSPD, Y, 50., 300.)

4.8.9. Wait Statement**WAIT**

The WAIT statement will hold until a specified condition is met. Four different wait conditions may be specified. The statement may wait until an axis is IN_RANGE, where it will wait for the axis to be at a position greater than the low position value and less than the high position. Both positions are specified in absolute user units. This mode requires the constant IN_RANGE to be specified.

If the range specified is very small (lower = upper) and the axis crosses thru the range quickly, then the wait may hang. Because the axis enters and exits the range before the position monitoring gets a chance to see it.



Variables may be used to specify the low and high position ranges. These variables are evaluated when the statement is executed. However, once evaluated, the values the U600/U631 monitor do not change, even if the value of the variables change.



The statement may wait until the axis it is slaved to is within a specified absolute position range, where it waits for the master axis to be at a position greater than the low position value and less than the high position. Both positions are specified in absolute user units. This mode requires the constant IN_RANGE_MASTER to be specified. The statement may wait until a cam table has been calculated, by specifying the number of the table. This mode requires the constant TABLE_READY to be specified.

The statement may also wait until a drive is enabled, by specifying the axis. This mode requires the constant DRIVE_ENABLED to be specified.

When the program is aborted, all wait conditions are terminated.



SYNTAX: (WAIT, IN_RANGE, axis, low_pos, high_pos)
 or
 (WAIT, IN_RANGE_MASTER, axis, low_pos, high_pos)
 or
 (WAIT, TABLE_READY, table_number)
 or
 (WAIT, DRIVE_ENABLED, axis)

EXAMPLE:

```
(WAIT, IN_RANGE, X, 30., 50.)
(WAIT, IN_RANGE_MASTER, X, -30., -10.)
(WAIT, TABLE_READY, 1)
(WAIT, DRIVE_ENABLED, Z)
```

4.8.10. The WTCH Statement

WTCH

The WTCH statement controls the specified virtual IO bit, setting it to a logic 1 if the axis position is in the specified position range, otherwise setting it to 0. The position range is specified in user units as a low and a high absolute position. The watches are active until they are cleared or the program is aborted.

All the WTCH statements may be cleared by using the WTCH statement and specifying the virtual-IO# as -1. WTCH statements may not be selectively cleared. When clearing the watches, the axis and range values are ignored, but must be specified. After watch statements are cleared, the specified I/O bit will remain as it was last set, it is not cleared.



Variables may be used to specify ranges, these variables are evaluated when the statement is executed. However, once evaluated, the values the U600/U631 watches do not change, even if the variables values have changed.



When the program is aborted, all watches become inactive.



If the specified range is very small (lower = upper) and the axis crosses thru the range quickly, the watch may never set the bit because the axis enters and exits the range before the position monitoring gets a chance to see it.

SYNTAX: (WTCH, axis, low_pos, high_pos, virtual-IO#)

EXAMPLE:

```
(WTCH, X, 10., 60., 9) ; set virtual bit 9 when in range
(WTCH, X, 10., 60., -1) ; clear ALL active WTCH statements
```

4.9. File Operation Commands

File operation commands permit the user to read and write data from various data files selected. The user can open a file, reset the file pointer to the beginning of the file and read or write the data desired. A file number is assigned to the file once it is opened. This file number is required for all reading, writing, resetting the file pointer, and closing of files. The user can read and write up to 30 values per line, from (and to) the selected data file.

4.9.1. File Open Command

FILEOPEN

The FILEOPEN command opens a user data file for reading or writing. As can be seen from the syntax diagram below, “*filespec*” is the name of the file to be opened. The program directory will be assumed by default if a path is not specified. “*Var*” is the file number assigned to the file selected by the user. If the filename begins with a numeral, an absolute filepath must be specified so that the “*filespec*” will not be interpreted as a numeric parameter. Mode refers to the mode of file access. Mode = 0 opens a new file even if an old one exists. In mode 0 the user can read or write the file. Mode = 1 opens an existing file for reading only. Mode = 2 opens an existing file only for appending records to the end of the file. Mode 0 is the default.

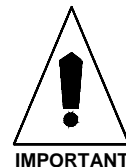
SYNTAX: (*FILEOPEN filespec = var [mode]*)

EXAMPLE:

Refer to FILEWRITE command for an example.

The programmer must ensure that the file is not already open when using a FILEOPEN in modes 0 or 2. If the file is currently open, the results are unpredictable.

If no directory is specified in the filename, the U31 programs directory is assumed.



4.9.2. FILEEOF Command

FEOF

The FEOF command determines whether the end of stream (file) has been reached. Once end-of-file is reached, FEOF returns “-1” unless the FILERESET command is executed. From the syntax below “*fp*” is a variable holding file index from FILEOPEN. “*Result*” is variable holding the return value from the command FEOF. The FEOF function returns a non zero value if not at end of file and returns “-1” if at end of file.

SYNTAX: (*FEOF fp result*)

EXAMPLE:

Refer to FILEWRITE command for an example.

4.9.3. File Read Command

FILEREAD

The FILEREAD command permits the user to read a line of selected data from the user data file into CNC variables. Looking at the syntax diagram below, “*var1*” is the filename associated with file that the user is reading from. “*Var2*” is the variable that will hold the first value on the line read from the file. “*Var3*” thru “*var30*” are the optional variables that will hold the other values on the line read from the file.

SYNTAX: (*FILEREAD var1 var2 [var3...var30]*)

EXAMPLE:

Refer to FILEWRITE command for an example.



FILEREAD can be used to read the text files not written by FILEWRITE.

The values in the file can be integer or floating point format and must be separated by one or more spaces.

4.9.4. File Close Command

FILECLOSE

The FILECLOSE command closes the file that the user opened. “*Var*” is the filename assigned to the file that the user would like to close.

SYNTAX: (*FILECLOSE var*)

EXAMPLE:

Refer to FILEWRITE command for an example.

4.9.5. File Reset Command

FILERESET

The FILERESET command resets the file pointer within the data file to the start of the file for reading or writing. “*Var*” is the filename assigned to the file that you would like to reset.

SYNTAX: (*FILERESET var*)

EXAMPLE:

Refer to FILEWRITE command for an example.



In mode 2, a file can be reset, but this has no effect. All data written in mode 2 is always appended to the end of a file.

4.9.6. File Write Command**FILEWRITE**

The FILEWRITE command permits the user to write up to 30 values to a specified data file followed by a new line. “Var” is the filename associated with the file to which the user is going to write. “Data” is the information to be written to the file and must be a list of variables.

The FILEWRITE command writes 7 digits past the decimal point for each value. Each value is separated by a space. The records written are always padded out to a fixed length with spaces.

SYNTAX: (*FILEWRITE Var data*)

EXAMPLE:

(DVAR fil,var1, var2, var3, var4, var5)	;declare the variables
(FILEOPEN test.dat = fil)	;open the file
var1=5 var2=4 var3=3.2 var4=6 var5=9	
(FILEWRITE fil var1 var2 var3 var4 var5)	;write to the file
var1=9 var2=6 var3=3.2 var4=4 var5=5	
(FILEWRITE fil var1 var2 var3 var4 var5)	;write to it some more
(FILERESET fil)	;file pointer is now at end of file
(FILECLOSE)	;reset file pointer back to start
(FILEOPEN test.dat = fil 1)	;open (read only)
(OPENCWD)	;open the customer display window
(FILEREAD fil var1 var2 var3 var4 var5)	;read the first line of data
(DISPLAY var1 var2 var3 var4 var5)	;display the first line of data
(FILEREAD fil var1 var2 var3 var4 var5)	;read the second line of data
(DISPLAY var1 var2 var3 var4 var5)	;display the second line of data
IF FILEEOF=0 THEN	;if at end of file
(FILERESET fil)	;reset to start
ENDIF	
(FILEREAD fil var1 var2 var3 var4 var5)	;reread first line of file
(DISPLAY var1 var2 var3 var4 var5)	;redisplay first line of file also
(CLOSECDW)	;close the customer display window
(FILECLOSE fil)	;close it

4.10. Master-slave Motion Commands

These CNC commands allow the CNC programmer to command master/slave motion. Master/slave motion is the most general form of motion allowing the user to command a single axis to move with virtually any position or velocity profile. Also, by syncing multiple slave axes' to a common master, the programmer can move multiple axes' in synchronized motion. However, the power of master/slave motion comes at a cost, there is more the programmer must do and understand to get the proper results. Master/slave motion was originally developed for grinding cams, electronic camming, but has many other uses such as making an axis follow a handwheel. So for historical reasons master/slave motion is often referred to as "camming".

Normally two axes are involved in master/slave motion. The axis that executes the desired motion is called the "slave" axis. The user must also provide a "master" axis, used to direct the slave. However, the master can be a virtual axis (refer to the *UNIDEX 631/U600* User's Manual for additional information on NULL feedback) and therefore need not physically exist. The programmer provides a table of coordinates (cam table) that specify slave axis positions given for a particular master axis position. Refer to Figure 4-4 for profile.

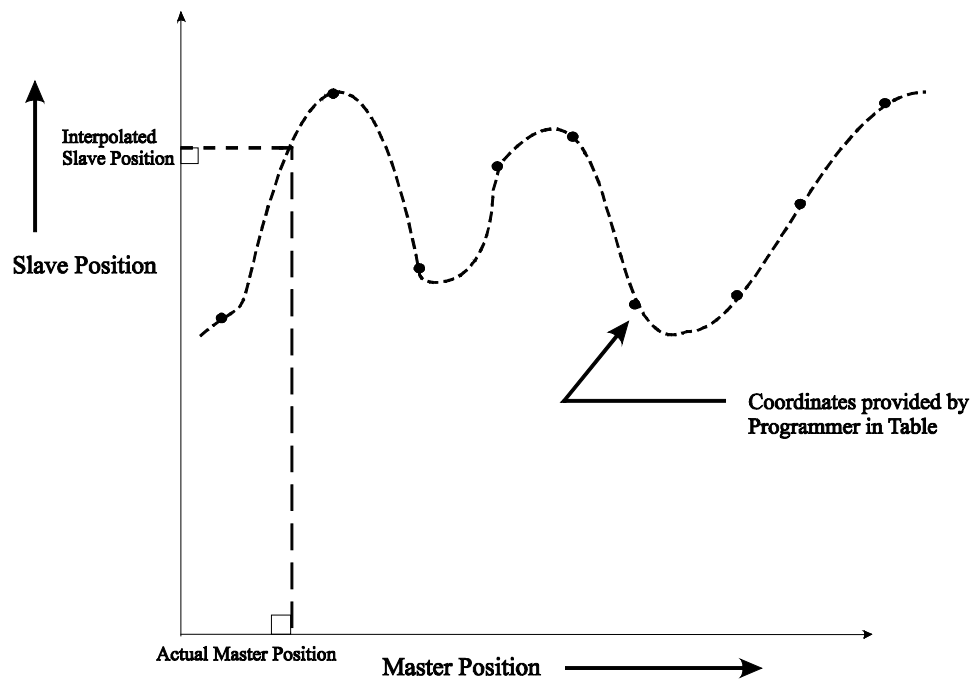


Figure 4-5. Master/Slave Profile

The programmer can also specify slave axis velocities for given master axis positions (see section 4.10.2 SYNC Command on mode 3).

4.10.1. CONFIGM Command**CONFIGM**

The CONFIGM command configures the master/slave relationship. This indicates to the slave axis to get its master coordinates from the master axis. The parameters may be axis names or numbers given. "Axis $spec1$ " is the master axis and "axis $spec2$ " is the slave axis.

SYNTAX: (CONFIGM axis $spec1$ axis $spec2$)

EXAMPLE:

(CONFIGM X Y)	;Configure the master/slave relationship
---------------	--

4.10.2. SYNC Command**SYNC**

This statement synchs a master to a slave. The synching will wait until all motion (if any is current) is done on the slave before synching. The user is cautioned that camming behavior can be complex and that the following description along with the description of the related parameters must be fully understood to produce the desired results.

Do not desynch from an axis in motion or the slave will stop abruptly. Use the *ENDM* command prior to desynching.



After a slave is synched to a master the slave's motion will follow the master axis's motion. After a slave is desynched the slave's motion no longer follows the master axis. The user must provide a synch mode as a parameter in the SYNC statement. The SYNC mode can be 0, 1, 2, or 3. A mode of 0 desynchs a slave axis from a cam table. Modes of 1, 2, and 3 synch the table up. To choose the correct mode for synching up the programmer must be cognizant of a number of complex details of the camming process as described below.

Modes 1 and 2 differ only in the behavior of the slave axis immediately upon synching up as described in the following paragraph. Note that mode 2 may cause slave axis motion in addition to the motion dictated by the camming. Using the following example: suppose that at the moment of synching, the master is currently at position "m" and the slave at position "x". Further, suppose that the table specifies that at master position "m" the slave should be at position "s". Mode 2 will direct the slave to move from position "x" to position "s" as the table synchs up. This mode 2 move is initiated simultaneously with the initiation of the camming move. Therefore, immediately after synching the total motion will be a sum of the motion directed by the camming and the motion directed by the mode 2 synch. At some later point in time the mode 2 sync move will be complete and the total motion will be determined by the camming. The mode 2 move is performed at the speed specified by the *SYNCSPEED* axis parameter and it is subject to normal acceleration and deceleration as dictated by the *ACEL* and *DECEL* axis parameters. Mode 1 will not cause any automatic movement of the slave. Instead the table is reinterpreted to mean that at master position "m" the slave must be at position "x". All following points in the table will be offset accordingly (x-s is effectively added to all slave position table coordinates).

An important note is that the program can resynch to a new table while the slave is already synched to a current table (without desynching in between with SYNC mode 0's).

However, modes 1 and 2 provide no protection for any jump in slave commanded positions that may be caused by the potentially different values in the two tables. The tables are switched instantaneously without any decel or accel.

Mode 3 is very different from modes 1 and 2. In mode 3 the slave values are interpreted as axis velocities, not axis positions. The table then controls slave velocity based on a master position.

MAXCAMACCEL - Mode 3 offers acceleration/deceleration protection that can be used when synching “on the fly” (without desynching in between with SYNC mode 0’s). If this parameter is non-zero, the slave axis will not exceed this acceleration while camming. This parameter is not used in modes 1 and 2. To deactivate this feature, set the parameter value to “0”.

CAMOFFSET - This parameter is added to the master position before doing the table lookup. For example, if the table covers master positions from 0 to 360 degrees the actual master position is 2 degrees and *CAMOFFSET* is 3 degrees, then CNC will use the value of 5 degrees as the master position to look up in the table with.

MASTERLENGTH - If upon synching the actual master position lies outside of the master coordinates in the table, the table master coordinates will be pushed up or down in units of *MASTERLENGTH* until it lies within the table. As an example, if the table covers from 0 to 359 degrees (the master is rotational) and the current master coordinate is 361 degrees the CNC will use the 1 degree entry in the table to direct the slave.



Camming is not intended to be used for nonrotational master axes’ when the current master position lies outside the master positions in the table.

The first parameter, “*axisspec*” is the slave axis, the second parameter “*parm*” is the table number, and the third parameter “*integ*” is the SYNC mode.

SYNTAX: (*SYNC axisspec parminteg*)

EXAMPLE:

(SYNC Y 3 2)	;Synchs the table up
(SYNC Y 3 0)	;Desynchs the table

4.10.3. FEDM Command

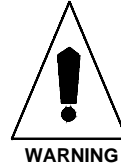
FEDM

The FEDM command is categorized as a camming or asynchronous motion command. Refer to section 4.11.1. for information on this command.

4.11. Asynchronous Motion Commands

This group of commands allow the user to execute motion commands that do not suspend the CNC program while the motion is occurring. The normal motion commands (G0 and G1) will always suspend the CNC program until the motion is complete. Asynchronous motion commands take virtually no time to execute. The motion is completed normally while the CNC continues executing more commands.

If an asynchronous motion is being executed and the program encounters another motion command on the same axis, the results will be unpredictable.



4.11.1. FEDM Command

FEDM

This statement executes motion on a single slave axis, stopping at the given target position. The motion is asynchronous to the CNC program execution, meaning the motion has begun and the next CNC statement is immediately executed. The FEDM command exhibits the same accel/decel behavior as a G1 or G0. This command is designed to be used while camming (it is applied to the motion in addition to a master/slave driving the given axis). The speed parameter (*param2*) is in unusual units; user units per second or degrees per second for a rotational axis. “*Axispec*” is the axis to feed in on (the slave) and “*param1*” is the amount to feed relative to the current position.

SYNTAX: *FEDM axispec param1param2*

EXAMPLE:

(FEDM Y 4.5 6.7) ;Infeed the slave to 4.5 inches, at 6.7 inches per second
--

4.11.2. Index Statement

INDEX

The INDEX statement initiates a relative move on a designated axis at a specified speed, then continues with the next program line without waiting for the index to finish. Distance is specified in user units and velocity is specified in user units/sec. The axis specified may be a soft or hard axis name.

SYNTAX: (*INDEX, axis, distance, speed*)

EXAMPLE:

(INDEX, X, 50., 300.)

4.11.3. Oscillate Command**OSC**

The OSCillate command causes a specified axis to oscillate (cycle) a specified distance at the specified velocity. The sign (+ or -) of the distance determines the initial direction of the move and is in user units. The feedrate is in user units per second. After this command is executed, the next command in the user's program immediately begins. To halt the axis a zero feedrate or distance can be specified in a subsequent use of this command.

SYNTAX: (*OSC, axis, dist, feed*)

Where

<i>axis</i>	-	is the axis to oscillate
<i>dist</i>	-	is the distance of the moves (sign determines initial direction)
<i>feed</i>	-	is the velocity the axis will move

4.11.4. MOVETO Statement**MOVETO**

The MOVETO statement initiates the move of an axis to a specified absolute position at a specific speed, then continuing with the next program line without waiting for the move to finish. The position is specified in user units, velocity is specified in user units/sec. The axis may be a soft or hard axis name.

SYNTAX: (*MOVETO, axis, position, speed*)

EXAMPLE:

(MOVETO, X, 50., 300.)

4.11.5. STRM Command

This statement begins motion on a single axis without stopping at any given target position. The motion is asynchronous to the CNC program execution, that is the motion is begun, and the next CNC statement is immediately executed. The speed parameter is in unusual units: user units per second. The *STRM* command accelerates instantaneously (jumping up to the commanded velocity). Use the *ENDM* command to end motion on an axis that had a *STRM* executed on it.

SYNTAX: (*STRM* <axis name> <constant> <var1>)

Where

<i>axis name</i>	- axis to feed in on (the slave)
<i>constant</i>	- direction to go (1 is positive, -1 negative)
<i>var1</i>	- speed to feed at (in user units per second or degrees per second for rotational axes)

▽ ▽ ▽

CHAPTER 5: OPTIONAL PSO COMMANDS

In This Section:	
• Introduction.....	5-1
• Programming Commands.....	5-2
• Conditional Tracking Based on Input States.....	5-3
• Position Synchronized Output Firing Distance Entry.....	5-6
• Enable/Disable Position Synchronized Output Firing.....	5-9
• Position Synchronized Output Using Bit Mapping.....	5-12
• Position Synchronized Output Pulse Configuration.....	5-14
• Position Synchronized Output with Real-time Control.....	5-16
• Digital/Analog Output Command.....	5-18

5.1. Introduction

The Position Synchronized Output Board provides a variety of outputs that may be used to synchronize control with motion. It is most commonly used for Laser firing. With the use of a variety of commands, the PSO Board may be instructed to activate up to four outputs with analog level controls and various types of single-shot or pulse train outputs. The Position Synchronized Output Board is activated through either a parts program or from the MDI Mode. The following sections provide the commands related to PSO function.

The PSO commands are not case sensitive, although throughout this chapter the PSO commands appear in uppercase letters for easy recognition.



This chapter uses the typographical conventions listed in Table 5-1.

Table 5-1. Conventions for this Section

Example	Description
<i>input_num</i>	Words in italic indicate information that you must supply to validate the command.
[option]	Items between brackets are optional.
PSOC, <i>mode</i> ,[{ <i>i,n</i> <i>in_map,out_map</i> }]	Braces and a vertical bar indicate a choice among two or more items. You must choose one of the items unless brackets surround the braces.
PSOT, <i>case,condition,condition2</i> ,...	Three dots following an item indicate that more items having the same form may be included.
⋮	A column of three dots indicates that part of an example program has been omitted.

5.2. Programming Commands

The PSO supports seven functional groups of programming commands. These functional groups of commands are listed in Table 5-2 and explained in detail in the sections that follow.

Table 5-2. PSO Programming Commands Summary

Command	Page	Description
PSOC	5-3	Conditional tracking based on input states
PSOD	5-6	Firing distance entry
PSOF	5-9	Specify tracking axes and begin tracking
PSOM	5-12	Bit mapping data download
PSOP	5-14	Laser pulse output definition
PSOR	5-16	Real time control of tracking
PSOT	5-18	Digital and analog output control

5.3. Conditional Tracking Based on Input States

PSOC

The PSOC command is used to enable unconditional tracking or enable conditional tracking based on the state(s) of up to 24 digital inputs.

PSOC 0 through 3 are not implemented at this time.



SYNTAX: **PSOC,mode** [{ *i,n* | *in_map,out_map*}]

5.3.1. MODE Arguments For PSOC

The mode argument defines one of four possible ways to use the PSOC command. The argument can range in value from 0 to 3 and the following sections describe their meaning.

5.3.1.1. Mode Argument - 0

Mode argument “0” enables position tracking unconditionally. Input signal conditions are ignored (Default). This mode has no additional arguments. Syntax for this mode is **PSOC,0**.

5.3.1.2. Mode Argument - 1

Mode argument “1” enables position tracking when a specified input number *i* (*i*=0-7) is in a specified state *n* (*n*=0 means a “low” state, *n*=1 means a “high” state). Syntax for this mode is **PSOC,1,*i,n*** (where *i* specifies the input number and *n* specifies the required state of the input for tracking). In this mode, counter data is retained when the position counter is disabled.

5.3.1.3. Mode Argument - 2

Mode argument “2” enables position tracking when a specified input number *i* (*i*=0-7) is in a specified state *n* (*n*=0 means a “low” state, *n*=1 means a “high” state). Syntax for this mode is **PSOC,2,*i,n*** (where *i* specifies the input number and *n* specifies the required state of the input for tracking). In this mode, counter data is reset to 0 when the position counter is disabled.

5.3.1.4. Mode Argument - 3

Mode argument “3” enables position tracking when input bits are configured as specified in the *in_map* argument (0 = “low”, 1 = “high”, and x = input bit is not checked). If the inputs are not configured as specified in the *in_map* argument (that is, the input condition is false), then the outputs are set according to the *out_map* argument. Syntax for this mode is **PSOC,3,*in_map,out_map***.

5.3.2. PSOC Arguments

The arguments used by the PSOC command vary based on the mode that is being used (refer to Section 5.3.1 Mode Arguments for PSOC). The following sections give a summary of all arguments used by the PSOC, *mode* command.

5.3.2.1. *i* Argument

The “*i*” argument specifies the single PSO input that controls the conditional tracking. This argument can range from 0 to 23. If the state of the input specified by argument “*i*” equals the pre-defined state “*n*”, then the condition is true and position tracking is enabled. This argument only used in modes 1 and 2 of the PSOC command.

5.3.2.2. *n* Argument

The “*n*” argument specifies the desired state for the selected input “*i*”. If the state of input “*i*” (0=low or 1=high) equals the desired state “*n*” (0=low or 1=high), then the condition is true and position tracking is enabled. This argument only used in modes 1 and 2 of the PSOC command.

5.3.2.3. *in_map* Argument

The “*in_map*” argument is a bitmap that defines the desired states of one or more selected inputs. If the actual states of the inputs match the desired states specified in the “*in_map*” argument, then the condition is true and position tracking is enabled. If this condition is false, then the second bitmap (*out_map*) defines the desired condition of the PSO outputs. This argument only used in mode 3 of the PSOC command.

5.3.2.4. *out_map* Argument

The “*out_map*” argument is a bitmap that defines the desired states of one or more selected outputs when the input condition is not met (when position tracking is disabled). When position tracking is not enabled (i.e., the actual states of the PSO inputs do not match the desired states specified in the “*in_map*” argument), then the PSO outputs are set according to “*out_map*”. This argument only used in mode 3 of the PSOC command.

5.3.2.5. *input* Argument

The “*input*” argument is the number of bytes assigned as inputs on the I/O bus. An assignment of “0” indicates that none of the lines will be configured as inputs and bits 0-23 will all be configured as outputs. An assignment of “1” indicates that bits 0-7 will be configured as inputs; “2” indicates bits 0-15 will be configured as inputs, and “3” indicates bits 0-23 will be configured as inputs. This argument only used in mode 4 of the PSOC command.

5.3.2.6. *output* Argument

The “*output*” argument is the number of bytes assigned as outputs on the I/O bus. An assignment of “0” indicates that none of the lines will be configured as outputs and bits 0-23 will all be configured as inputs. An assignment of “1” indicates that bits 16-23 will be configured as outputs; “2” indicates bits 8-23 will be configured as outputs, and “3” indicates bits 0-23 will be configured as outputs. This argument only used in mode 4 of the PSOC command.

Each bit of the *in_map* and *out_map* bitmap arguments is configured using a “0”, “1” or “x”. A “0” indicates a low input/output and a “1” indicates a high input/output. For *in_map*, “x” indicates that the associated input state should not be checked. For *out_map*, “x” indicates that the associated output state remains unchanged.



EXAMPLE:

```

:                                     ;Motion controller pre-processing
(PSOC,3,xx1x0101,xxxx11000011xxxx) ;Conditional PSO tracking enable
:                                     ;Motion commands, post-processing, etc.
    
```

In this example, tracking is enabled only when inputs 0, 2 and 5 are high, and when inputs 1 and 3 are low. The states of the other inputs are ignored. If tracking is not enabled (i.e., when the inputs do not match the above criteria), then the PSO outputs are set according to the final argument. In this case, outputs 4, 5, 10 and 11 are driven high, and outputs 6, 7, 8 and 9 are driven low. All other outputs are unaffected.

5.4. Position Synchronized Output Firing Distance Entry PSOD

The PSOD command specifies the number of machine steps to be traveled before output synchronization occurs. Distances may be entered individually or sequentially through the use of the array variables. This command is only used in conjunction with the PSOF,3 command.

SYNTAX: PSOD,*mode* {,*distance* | ,*array[x]*, $\pm m$ }

5.4.1. MODE Arguments For PSOD

The mode argument defines one of three possible ways to use the PSOD command. The argument can range in value from 0 to 2 and the following sections describe their meaning.

5.4.1.1. Mode Argument -0

The mode argument “0” indicates the pulse output will occur at a fixed incremental distance *distance*.

5.4.1.2. Mode Argument - 1

The mode argument “1” indicates the pulse output will occur at incremental distances as defined in the user “*array[x]*”. The user “*array[x]*” will either increment (+) or decrement (-) to retrieve a total of “*m*” values. For example, the command PSOD,1,*array*[3],10 will define 10 firing distances. The value of the first firing distance is expected to be in *array*[3], the second in *array*[4] and so on to *array*[12] (a total of 10 firing distances). Syntax for this MODE is PSOD,1,*array*[*x*], $\pm m$.

5.4.1.3. Mode Argument - 2

Mode argument “2” indicates the pulse output will occur at absolute distances as defined in “*array[x]*” of the user CNC program and will either increment (+) or decrement (-) to retrieve a total of “*m*” values. For example, the command PSOD,2,*array*[6],5 will define 5 firing distances. The value of the first firing distance is expected to be in *array*[6], the second in *array*[7] and so on to *array*[10] (a total of 5 firing distances). Syntax for this mode is PSOD,2,*array*[*x*], $\pm m$.



The $\pm m$ argument assumes a positive direction if no sign is specified.

5.4.2. PSOD Arguments

The arguments used by the PSOD command vary based on the mode that is being used (refer Section 5.4.1 Mode Arguments for PSOD). The following sections give a summary of all arguments used by the PSOD,mode command.

5.4.2.1. *distance* Argument

The “*distance*” argument specifies the fixed incremental firing distance in machine steps at which the pulsed output occurs. This distance must be less than 2^{23} machine steps and is determined differently depending on how many axes are involved. Refer to Table 5-3.

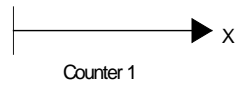
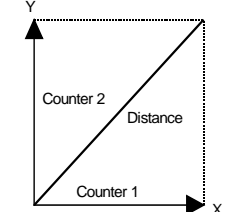
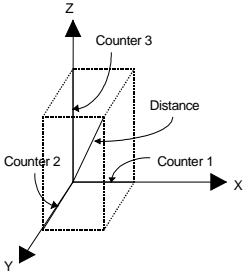
5.4.2.2. *array[x]* Argument

The “*array[x]*” argument specifies the starting array variable from the UNIDEX 31/U600 that contains the first distance datum. This argument is used in conjunction with the “m” argument.

5.4.2.3. *±m* Argument

The “*±m*” argument is used to specify the number of times to increment or decrement (based on the sign) sequentially through the array (from the starting array index “*array[x]*”) to get additional firing distances. This argument represents the total number of elements to use (including *array[x]*) and can range from ± 2 to ± 255 . For example, the command PSOD,2,array[4],-5 gets the first distance datum from array[4]. Subsequent distance values are read sequentially from array[3] through array[10].

Table 5-3. Distance Calculations for Multiple Axes Using the PSOD Command

Number of Axes	Distance Calculation	Diagram
1	Distance = Counter1	
2	Distance = $\sqrt{\text{Counter1}^2 + \text{Counter2}^2}$	
3	Distance = $\sqrt{\text{Counter1}^2 + \text{Counter2}^2 + \text{Counter3}^2}$	

EXAMPLE:

(PSOD,0,5)	;Pulsed output occurs every 5 machine steps
(PSOP,4,10)	;10 μsec pulse output on position
(PSOF,3,X,Y)	;Activate output firing pulse train
(PSOD,1,array[199],-200)	;Pulsed output occurs at incremental distances found in ;array[199] thru array[0] (200 total firing distances)
(PSOF,3,X,Y)	;Activate output firing pulse train
(PSOD,2,array[100],50)	;Pulsed output occurs at absolute distances found in array[100] ;thru array[149] (50 total firing distances)
(PSOF,3,X,Y)	;Activate output firing pulse train

5.5. Enable/Disable Position Synchronized Output Firing **PSOF**

The PSOF command activate or deactivates the pulse train output and tracking features.

SYNTAX:

PSOF,mode [{,num | , axis1 [,axis2 [,axis3]]] | ,±dist ,axis1 [,axis2 [,axis3]] }

5.5.1. MODE Arguments For PSOF

The mode argument defines one of six possible ways to use the PSOF command. The argument can range in value from 0 to 5 and the following sections describe their meanings.

5.5.1.1. Mode Argument - 0

The mode argument “0” indicates the output firing pulse train and tracking features are disabled. (default). Syntax for this mode is **PSOF,0**.

5.5.1.2. Mode Argument - 1

Mode argument “1” activates the output firing pulse train (as established by the PSOP command). The pulse train will continue until it is disabled by the PSOF,0 command. No position tracking occurs in this mode. Syntax for this mode is **PSOF,1**.

5.5.1.3. Mode argument - 2

Mode argument “2” activates the output firing pulse train (as established by the PSOP command) for “num” number of times. If *num*=0, then the output firing pulse train will not be activated until the previous output firing pulse train is complete. No position tracking occurs in this mode. Syntax for this mode is **PSOF,2,num**.

5.5.1.4. Mode Argument - 3

Mode argument “3” activates the output firing pulse train (as established by the PSOP command). The position counter locks on to the motion of the specified. Up to three axes may be “locked on” simultaneously. Output firing occurs at distances established by the PSOD command. Syntax for this mode is **PSOF,3,axis1[,axis2,axis3]**.

5.5.1.5. Mode Argument - 4

Mode argument “4” activates the output firing pulse train and locks the position counters onto the specified axes (e.g., X, Y, Z). Up to three axes may be “locked on” simultaneously. The output firing pattern is determined by bit mapping as established by the PSOM command. The bit values serve the following functions:

- 0 - Causes the output to go/remain low
- 1 - Causes the output to go/remain high.

The pulse output occurs at a fixed incremental distance “*dist*”. If “*dist*” is positive (no sign or “+”), the bit pattern is run in a forward direction. If “*dist*” is negative (“-”), the bit pattern is run in reverse. Syntax for this mode is **PSOF,4, $\pm dist$,axis1,axis2,axis3**.

5.5.1.6. Mode Argument - 5

Mode argument “5” activates the output firing pulse train and locks the position counters onto the specified axes (e.g., X, Y, Z). Up to three axes may be “locked on” simultaneously. The output firing pattern is determined by bit mapping as established by the PSOM command. The bit values serve the following function:

- 0 - Causes no output
- 1 - Causes the output to be a single pulse train, defined by the PSOP command.

The pulse output occurs at a fixed incremental distance “*dist*”. If “*dist*” is positive (no sign or “+”), the bit pattern is run in a forward direction. If “*dist*” is negative (“-”), the bit pattern is run in reverse. Syntax for this mode is **PSOF,5,*dist*,axis1,axis2,axis3**.



The $\pm dist$ argument assumes a positive distance if no sign is specified.

5.5.2. PSOF Arguments

The arguments used by the PSOF command vary based on the mode that is being used (refer to Section 5.5.1. Mode Arguments for PSOF). Some forms of the PSOF,*mode* command have no additional arguments. The following sections give a summary of all arguments used by the PSOF,*mode* command.

5.5.2.1. *num* Argument

The “*num*” argument specifies the number of times to activate the output firing pulse train (as established by the PSOP command). This argument is used only by the PSOF,2 command.

5.5.2.2. axis# Argument

The “axis#” arguments specify which axes (up to 3) are to be locked on to the position counters. These arguments are specified using the axis names X, Y or Z. These arguments are used exclusively by commands PSOF,3, PSOF,4 and PSOF,5.

5.5.2.3. ±dist Argument

The “±dist” argument specifies the fixed incremental distance in machine counts at which the pulsed output occurs. If “dist” is positive (no sign or “+”), then the bit pattern will be run in a forward direction. If “dist” is negative (“-”), then the bit pattern will be run in the reverse direction. This distance argument is used only in commands PSOF,4 and PSOF,5. The value for this argument is determined differently depending on how many axes are involved. Refer back to Table 5-3.

EXAMPLE:

⋮	;Motion controller pre-processing
(PSOF,0)	;Output firing pulse train and tracking disabled (default)
⋮	;Motion controller pre-processing
(PSOP,1,5,10,0)	;Define pulse output train
(PSOF,1)	;Activate the pulse train and continue until disabled
(PSOP,1,5,10,0)	;Define pulse output train
(PSOF,2,25)	;Activate the pulse train and continue 25 times
⋮	;Motion commands, post-processing, etc.
(PSOD,0,50)	;Define pulsed output (firing occurs every 50 machine steps)
(PSOP,1,5,10,0)	;Define pulse output train
(PSOF,3,X,Y)	;Activate the output firing pulse train and lock onto axes X and Y
⋮	;Motion commands, post-processing, etc.
⋮	;Motion controller pre-processing
(PSOM,0,array[0],64)	;Define output bit mapping (in array[0] thru array[15])
(PSOF,4,X,Y,Z)	;Activate output firing pulse train and lock onto axes X, Y and Z
⋮	;Motion commands, post-processing, etc.
⋮	;Motion controller pre-processing
(PSOM,0,array[0],64)	;Define output firing pattern (in array[0] thru array[15])
(PSOF,5,-500,X)	;Activate output firing pulse train, lock onto axis X, and run bit ;pattern in the reverse direction
⋮	;Motion commands, post-processing, etc.

5.6. Position Synchronized Output Using Bit Mapping PSOM

The PSOM command defines the characteristics of a bit pattern (stored in a programmable range of UNIDEX 631/U600 variables). Bit patterns are used to specify when the laser is to be “On” (PSOF,4) or when to fire a pulse output (PSOF,5).



PSOM 0 has not been implemented at this time.

SYNTAX: **PSOM,0,array[x],±size**

5.6.1. array[x] Argument

The “array[x]” argument specifies the array and starting index of the UNIDEX 31 variable that contains the starting number of the first thirty- two bit firing pattern.

5.6.2. ±size Argument

The “±size” argument specifies the total number of bytes (i.e., groups of 8 bits including “array[x]”) that make up the desired output firing bit map pattern. If “size” is positive (no sign or “+”), then the bit pattern continues from the most significant bit of the 32-bit starting bit pattern (specified in “array[x]”) to the least significant bit, and then continues forward (to the next sequential element of “array[x+1]”, always starting at the most significant bit) for “size” bytes. If “size” is negative (“-”), then the bit pattern continues from the least significant bit of the 32-bit starting bit pattern (specified in “array[x]”) to the most significant bit, and then continue in reverse (to the previous variable number “array[x-1]”, always starting at the least significant bit) for “size” bytes. This argument can range from ±2 to ±255. An array element holds the equivalent of four bytes of information (or 32 bits of firing points). Refer to Figure 5-1.

EXAMPLE:

⋮	;Motion controller pre-processing
(PSOM,0,array[0],64)	;Define output bit mapping (in array[0] thru array[15])
(PSOF,4,500X,Y,Z)	;Activate output firing pulse train and lock onto axes X, Y & Z
⋮	;Motion commands, post-processing, etc.
⋮	;Motion controller pre-processing
(PSOM,0,array[31],-128)	;Define output firing pattern (array[31] thru array[0])
(PSOF,5,-500,X)	;Activate output firing pulse train, lock onto axis X, and run bit pattern in the reverse direction
⋮	;Motion commands, post-processing, etc.

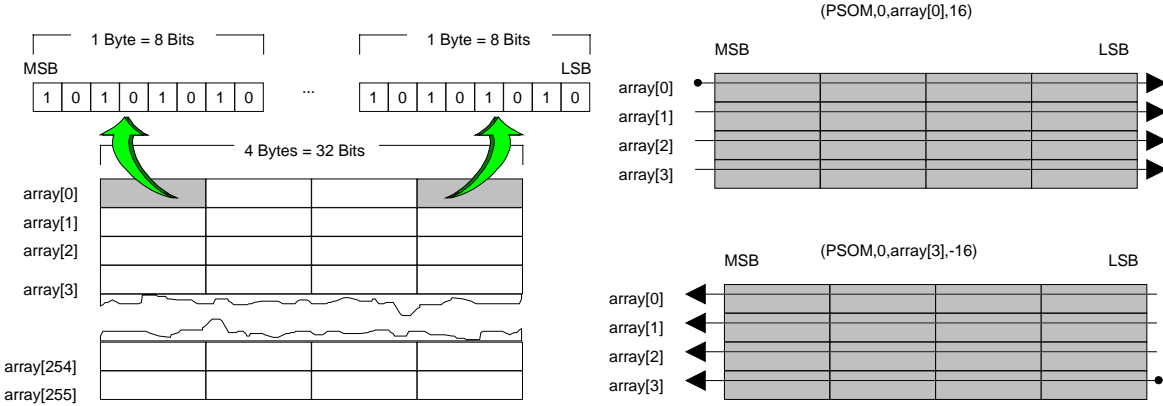


Figure 5-1. Bit Mapping Scan Patterns for PSOM Commands

5.7. Position Synchronized Output Pulse Configuration PSOP

The PSOP command configures the pulse output train.



PSOP 5 and PSOP 3 have not been implemented at this time.

SYNTAX: **PSOP**,*mode* { ,*w* | ,*l,w,t* | ,*l,w,t,r,g* | ,*array[x]*,*±m* }

5.7.1. MODE Arguments For PSOP

The mode argument defines one of six possible ways to use the PSOP command. The argument can range in value from 0 to 5 and the following sections describe their meaning.

5.7.1.1. Mode Argument - 0

Mode argument “0” sets the width *w* of a single pulse output in tenths of milliseconds. The syntax for this mode is **PSOP,0,w**.

5.7.1.2. Mode Argument - 1

Mode argument “1” sets a single pulse with definable pulse lead (*l*), pulse width (*w*) and pulse trail (*t*) characteristics. The pulse lead, width and trail arguments are specified in tenths of milliseconds. Syntax for this mode is **PSOP,1,l,w,f**.

5.7.1.3. Mode Argument - 2

Mode argument “2” sets a pulse train with definable pulse lead (*l*), pulse width (*w*), pulse trail (*t*), ramp up/down (*r*) and pulse gap interval (*g*) characteristics. The units used by the arguments are tenths of milliseconds. The pulse width starts at “r” and increments by “r” units until the pulse width reaches “w”. At this point, the pulse widths begin to decrease (in increments of “r” units) to a width of “r” units. The pulse gaps can be a fixed width (*g*>0) or can be made to ramp (i.e., match the size of the pulse width) (*g*=0). Syntax for this mode is **PSOP,2,l,w,f,r,g**.

5.7.1.4. Mode Argument - 3

Mode argument “3” sets a pulse train by defining a series pulse widths and gap widths (in tenths of milliseconds) in a series array elements (e.g., array[0] thru array[255]). The argument “*array[x]*” specifies the array element that contains the first gap width (in tenths of milliseconds). The next sequential array element contains the first pulse width (in tenths of milliseconds). Syntax for this mode is **PSOP,3,array[x],±m**.

5.7.1.5. Mode Argument - 4

Mode argument “4” sets the width “w” of a single pulse output in microseconds (with a minimum value of 1 microsecond). Syntax for this mode is **PSOP,4,w**.

5.7.1.6. Mode Argument - 5

Mode argument “5” sets output toggle mode. For example, if the pulse output is configured to occur at a fixed incremental distance of 500 machine steps (PSOD,0,500), then each 500-step motion is considered to be an event. After the first 500 steps, the pulse output is enabled. After the second 500 steps, the pulse output is disabled (i.e., odd-numbered events enable the pulse output, and even-numbered events disabled the pulse output). Syntax for this mode is **PSOP,5**.

5.7.2. PSOP Arguments

The arguments used by the PSOP command vary based on the mode that is being used (refer to Section 5.7.1. Mode Arguments for PSOP). Some forms of the *PSOP,mode* command have no additional arguments. The following sections give a summary of all arguments used by the *PSOP,mode* command.

5.7.2.1. *l* Argument

The “*l*” argument specifies the pulse lead in tenths of milliseconds. This argument only used with commands PSOP,1 and PSOP,2.

5.7.2.2. *w* Argument

The “*w*” argument specifies a pulse width in tenths of milliseconds. This argument is only used with commands PSOP,0 through PSOP,2 and PSOP4.

5.7.2.3. *t* Argument

The “*t*” argument specifies the pulse trail in tenths of milliseconds. This argument only used with commands PSOP,1 and PSOP2.

5.7.2.4. *r* Argument

The “*r*” argument specifies a ramp up/down time in tenths of milliseconds. This argument is used only with command PSOP,2. This argument allows the pulse width (and therefore the power output of the laser) to vary (i.e., increment then decrement) from “*r*” tenths of milliseconds to “*w*” (the full pulse width) in increments of “ $\frac{r}{w}$ ” tenths of milliseconds, and then back again. This feature is useful in applications involving laser drilling. In such applications, a pulse train that begins with small-width pulses (low power) and gradually increases to continuous constant-width output pulses (full power, for example), may be preferred over a steady pulse train of full-power pulses.

5.7.2.5. *g* Argument

The “*g*” argument specifies the pulse gap interval (between ramps) in tenths of milliseconds. These gaps can be variable-width based on the pulse train width “*w*” ($g=0$) or fixed width ($g>0$). This argument is only used with command PSOP,2.

5.7.2.6. *array[x]* Argument

The “*array[x]*” argument specifies the UNIDEX 31 array element that contains the “off/on” pulse train timing characteristic data in tenths of milliseconds. The specified variable contains the number of “off” tenths of milliseconds, the second sequential array element contains the number of “on” tenths of milliseconds, etc. for a total of “*m*” values.

5.7.2.7. $\pm m$ Argument

The “ $\pm m$ ” argument specifies the number of changes (i.e., on-to-off or off-to-on) that make up the desired output train. If “*m*” is positive (no sign or “+”), then the PSO gets sequential pulse/gap width data from increasing sequential array elements starting with “*array[x]*”. If “*m*” is negative (“-”), then the PSO gets sequential pulse/gap width data from decreasing sequential direct variables starting with “*array[x]*”.



The $\pm m$ argument assumes a positive distance if no sign is specified.

EXAMPLE:

```

:           ;Motion controller pre-processing
(PSOP,0,100) ;Define a single pulse output (For PSOP,0,100, pulse width = 10.0 ms)
(PSOD,0,5000) ;Define pulsed output (firing occurs every 5,000 machine steps)
:           ;Motion controller, post processing, etc.
(PSOP,1,50,10,10) ;Define pulse output train
(PSOF,2,25) ;Activate the pulse train and continue 25 times
:           ;Motion commands, post-processing, etc.
(PSOP,1,5,10,0) ;Define pulse output train
(PSOF,3,X,Y) ;Activate the output firing pulse train and lock onto axes X and Y
:           ;Motion commands, post-processing etc.

```

5.8. Position Synchronized Output with Real-time Control PSOR

The PSOR command provides various configurations of the PSO's position counter.

PSOR 0, 1, and 3 have not been implemented at this time.



SYNTAX: PSOR,*mode*

5.8.1. MODE Arguments For PSOR

The mode argument defines one of three possible configurations (0, 1 or 3) for the position counters. There are no additional arguments for this command. The following sections describe these modes.

5.8.1.1. Mode Argument - 0

Mode argument "0" clears all previous real-time control data from the counter. Syntax for this mode is **PSOR,0**.

5.8.1.2. Mode Argument - 1

Mode argument "1" stops the position counter from recording new data and retains current data under operator command. Syntax for this mode is **PSOR,1**.

5.8.1.3. Mode Argument - 3

Mode argument "3" stops the position counter from recording new data and returns the counter to zero. Syntax for this mode is **PSOR,3**.

5.9. Digital/Analog Output Command

PSOT

The PSOT command is used to set/clear digital output lines and/or set the desired voltage (-10.0 V to 10.0 V) of the analog outputs.

SYNTAX:

PSOT,*mode*{*,bit#*,*state*,... | *,states* | *,dac#*,*voltage*,... | *,dac#*,*v0*,*vmax*,*velocity* | *,dac#*,*v0*,*vmax*,*position* }

5.9.1. MODE Arguments For PSOT

The mode argument defines one of six possible configurations (0, 1, 2, 4, 6, or 8) for setting digital and analog outputs. The following sections describe these modes.

5.9.1.1. Mode Argument - 0

This mode is used to set individual digital output lines (argument “bit#”) to either a high signal (*state*=1) or a low signal (*state*=0). Syntax for this mode is **PSOT,0,bit#,state**,...

5.9.1.2. Mode Argument - 1

This mode is used to set a group of digital output lines high or low using a hexadecimal or decimal number specified by the “states” argument. Syntax for this mode is **PSOT,1,states**.

5.9.1.3. Mode Argument - 2

This mode is used to set the output voltage of the 4 D/A’s to a constant value. DAC output voltages can range from -10.0 V to 10.0 V and have a minimum step size of 0.3 mV. Syntax for this mode is **PSOT,2,dac#,voltage**.

5.9.1.4. Mode Argument - 4

This mode is used to set the output voltage of the desired DAC (0 to 3) to a value that is proportional to the velocity (velocity ramping). DAC output voltages can range from a programmable minimum value (at zero velocity) specified by argument *v0* to a maximum voltage (at target velocity *velocity*) specified by argument *vmax*. Syntax for this mode is **PSOT,4,dac#,v0,vmax,velocity**.



The zero velocity and target velocity are specified in machine steps per millisecond.

5.9.1.5. Mode Argument - 6

This mode is used to set the output voltage of the desired DAC (0 to 3) to a value that is proportional to the position (position ramping). DAC output voltages can range from a programmable minimum value (at initial position) specified by argument *v0* to a maximum voltage (at target position *position*) specified by argument *vmax*. Syntax for this mode is **PSOT,6,dac#,v0,vmax,position**.

5.9.1.6. Mode Argument - 8

This mode is used to vary the laser firing output (not the DAC outputs). It generates a laser firing output pulse at zero velocity as defined by the minimum on and off time parameters. Also, it decreases the off time of the pulse to the value specified by the minimum off time parameter as the velocity increases to the value designated by the velocity parameter.

Syntax for this mode is **PSOT, 8, on_time, off_time, min_off_time, vel**.

5.9.1.7. on_time# Argument

The “*on_time*” argument is specified in 100 μ sec increments, (i.e., *on_time* = 1 spc is equal to 100 μ sec on time), this is the on time at zero velocity.

5.9.1.8. off_time# Argument

The “*off_time*” argument is specified in 100 μ sec increments, (i.e., *off_time* = 1 spc is equal to 100 μ sec off time), this is the off time at zero velocity.

5.9.1.9. min_off_time# Argument

The “*min_off_time*” argument is the on time reached (decreased to) at the specified velocity. The “*min_off_time*” is specified in 100 μ sec increments (i.e., *min_off_time* = 1 is equal to 100 μ sec minimum off time).

5.9.1.10. vel# Argument

The “*vel*” argument is specified in counts per msec., which is the velocity where the minimum off time of the laser firing pulse is reached.

5.9.2. PSOT Arguments

The arguments used by the PSOT command vary based on the mode that is being used (refer to Section 5.9.1. Mode Arguments for PSOT). The following sections give a summary of all arguments used by the PSOT,*mode* command.

5.9.2.1. *bit#* Argument

The “*bit#*” argument specifies an individual bit number that corresponds to one of the 24 digital outputs. This argument is only used by the command PSOT,0.

5.9.2.2. *state* Argument

The “*state*” argument specifies the digital state (0=low, 1=high) of the previous output line (specified by the “*bit#*” argument). This argument is used by the PSOT,0 and PSOT,1 commands. Multiple groups of bit numbers and states may be specified.

5.9.2.3. *states* Argument

The “*states*” argument specifies a single hexadecimal or decimal number that defines the desired output states of the digital outputs. For example, the programming command PSOT,1,#A3C1 would set the output bits as follows: 1010 0011 1100 0001 (1010=A, 0011=3, 1100=C and 0001=1). The “*states*” argument may also be in decimal format (41921).

5.9.2.4. *dac#* Argument

The “*dac#*” argument specifies which of the four digital-to-analog converters (0 to 3) you want to set. This argument is only used by commands PSOT,2, PSOT,4 and PSOT,6.

5.9.2.5. *voltage* Argument

The “*voltage*” argument defines the desired output voltage (-10.0 to 10.0 volts) of the specified DAC channel (0 to 3). DAC channels and their respective voltages may be listed in the same PSOT command (e.g., PSOT,2,0,5,1,-5). This argument is only used by command PSOT,2.

5.9.2.6. *v0* Argument

The “*v0*” argument specifies the zero-state analog output voltage for proportional output modes PSOT,4 (*v0* is the analog output voltage at zero velocity) and PSOT,6 (“*v0*” is the analog output voltage at the initial position). This argument can range from -10.0 volts to 10.0 volts and is used only by modes PSOT,4 and PSOT,6.

5.9.2.7. *vmax* Argument

The “*vmax*” argument specifies the maximum analog output voltage at target velocity “*velocity*” in mode PSOT,4 or at the target position “*position*” in mode PSOT,6. This argument can range from -10.0 volts to 10.0 volts and is used only by modes PSOT,4 and PSOT,6.

5.9.2.8. *velocity* Argument

The “*velocity*” argument defines the target velocity (in machine steps per millisecond) at which the analog output defined by *dac#* will be at its maximum (as defined by “*vmax*”). Velocity can range from -2^{23} to $2^{23}-1$ machine steps per millisecond. This argument is used only in mode PSOT,4.

5.9.2.9. *position* Argument

The “*position*” argument defines the target position (in machine steps) at which the analog output defined in “*dac#*” will be at its maximum (as defined by “*vmax*”). “*Position*” can range from -2^{23} to $2^{23}-1$ machine steps. This argument is used only in mode PSOT,6.

Programming an output bit to a logical “1” state results in the corresponding pin being pulled to ground. A programmed “0” state results in a high impedance state on the corresponding output pin. During a reset, all outputs go to the high impedance state. All input lines are pulled to +5V by a 10KΩ resistor.



EXAMPLE:

```
(PSOT,0,0,1,1,1,2,1,3,1) ;Sets digital output lines OUT0-OUT3 high.
(PSOT,1,#4AB9)           ;Sets output lines to 0100 1010 1011 1001 (1=high, 0=low)
(PSOT,2,0,2.5)           ;Sets DAC0 (AOUT1) to 2.5 V.
(PSOT,4,0,3,7,125)      ;Sets DAC0 (AOUT1) for velocity ramping ;AOUT1 ranges
                        ;proportionally from 3 V to 7 V as velocity ranges from 0 to
                        ;125 machine steps/ms.
(PSOP,5)                 ;Toggle mode laser firing
array[0] = 300           ;Laser off for 300 machine steps
array[1] = 700           ;Laser on for 700 machine steps
(PSOD,1,array[0],2)      ;Incremental firing distance with 2 elements (V0 and V1)
(PSOF,3,X,Y)            ;Enable firing, track vector velocity in the X-Y plane
:                        ;Motion commands, post-processing, etc.
```



APPENDIX A: AXIS PARAMETERS

<p>In This Section:</p> <ul style="list-style-type: none"> Description A-1
--

Description

The following section provides a quick reference of axis parameters used by the UNIDEX 631/U600.

POS

This parameter specifies the observed position for a selected axis in machine steps taking into account any correction steps specified by an axis correction table. This parameter is a 32 bit signed integer having a range of approximately $\pm 2.1m (2^{31}-1)$. Upon initialization, the system sets the current observed position equal to the POS parameter value.

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.



RAWPOS

Same as "POS", except it does not reflect any axis correction changes.

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.



ECHO

This parameter allows you to set a "dummy" parameter. It has no effect on the operation of the controller, but may be used to test communications with the axis processor, or as a temporary holding area.

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.



CLOCK

This parameter allows you to set the starting count of the clock which has one millisecond resolution. The clock starts counting from the specified time. This is available for application programs that require a general purpose timer. Use of this parameter has no effect on the operation of MAINMENU.



This parameter can not be set from the Axis Parameter screen. Use the “SETPARAM” command in a program or manual mode to set this parameter.

AVGVELTIME

The axis processor card of UNIDEX 631/U600 maintains a read-only parameter called AVGVEL that reports the average velocity for a given axis. This average has no effect on the operation of the controller, but is maintained for the benefit of the application program. This parameter, AVGVELTIME, specifies the time period to average the velocity over. The units for this parameter are in milliseconds and must range from 10 to 1,000 milliseconds (in 10msec increments).

KI

This parameter sets the integral gain of the velocity loop for the selected axis. Refer to the *UNIDEX 631/U600 Hardware Manual* for a description of how this parameter functions in the servo loop. The valid range of this parameter is 0 to 10,000. The default value is 2,000.

KP

This parameter sets the proportional gain of the velocity loop for the selected axis. Refer to the *UNIDEX 631/U600 Hardware Manual* for a description of how this parameter functions in the servo loop. The valid range of this parameter is 0 to 100,000. The default value is 10,000.

PGAIN

This parameter determines the position gain of the position loop for the selected axis. Refer to the *UNIDEX 631/U600 Hardware Manual* for a description of how this parameter functions in the servo loop. The valid range of this parameter is 0 to 1,000. The default value is 10.

VFF

This parameter enables or disables the Velocity Feed Forward function. Once enabled, this function minimizes the position following error. A one enables this function, while a zero disables it.

DRIVE

This parameter enables and disables the motor's torque associated with an axis. A zero disables the drive, while a one enables it.

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.

**AUX**

This parameter controls the auxiliary output of a selected axis. A one asserts the output, while a zero de-asserts it.

Typically, this output may be used to activate a motor brake. The user may configure the FAULTMASK and AUXMASK parameters to cause this output to change state on an axis fault.

Therefore, each time a fault condition occurs, the system would apply a brake to the motor.

AFFGAIN

This parameter sets the Acceleration Feed Forward Gain used in the acceleration loop of the selected axis. The range for this parameter is -1,000,000 to 1,000,000. The default value is 0.

BLOCKMOTION

This parameter causes the axis to ignore any motion commands. The only exception is when the axis is currently under the control of a sync table. While the system blocks motion, the axis accepts commands to stop. A value of zero causes the system to process motion commands, while a one causes the system to ignore them.

There are many reasons why you may desire to use this feature. For example, if an axis must remain stationary throughout a process, an application program may temporarily block axis motion to prevent another concurrently executing application from initiating motion on that axis.

REVERSALMODE

To provide greater positioning accuracy, this parameter allows you to specify the number of machine steps required to compensate for any backlash present in the system. Backlash occurs when the ball screw changes direction and moves a fixed distance before the stage begins to move in the new direction. This parameter specifies the distance in machine steps. The valid range for this parameter is 0 to 1,000. The default is for zero backlash compensation (0 machine steps).

IMAX

This parameter sets the peak commanded output current. This is done by limiting the maximum output voltage of the current command signal, which is in turn translated into a proportional motor current by the drive module.

The range of this parameter is 0 to 32,767, where 10 volts is represented by the value 32,767. Determine the maximum input command voltage that your amplifier requires to produce the maximum desired motor current.

$$\text{Parameter Value} = \frac{\text{drive module max. input voltage}}{10} * 32767$$

The default value of this parameter is 32,767, producing a 10 volt current command signal, which would command the maximum current from the drive module.

IAVGLIMIT

This parameter detects an over current condition based upon on the setting of the IAVGTIME parameter. The value specified in the IAVGTIME parameter determines what time period to average the instantaneous current. An RMS current limit fault occurs if the RMS average exceeds the limit set by this parameter.

As with the IMAX parameter, the range of this parameter is 0 to 32,767, where 10 volts is represented by the value 32,767. Determine the maximum input command voltage that your amplifier requires to produce the maximum desired motor current.

$$\text{Parameter Value} = \frac{\text{drive module max. input voltage}}{10} * 32767$$

The default value of this parameter is 32,767, producing a 10 volt current command signal, which would command the maximum current from the drive module.

IAVGTIME

This parameter defines the time period over which the system will average the instantaneous current. The IAVG limit parameter is dependent on the setting of this parameter to detect an over-current condition. The unit of measure for this parameter is milliseconds and can range from 10 to 4,000 (msec). The default is 10 (msec).

POSERRLIMIT

This parameter determines the maximum position error that can occur on an axis before it generates a position error limit fault. The unit of measure for this parameter is machine steps, and can range between 0 and 10,000,000. The default value is 65,535 counts.

INPOSLIMIT

This parameter allows you to define the in-position band. If the axis has completed its move and the observed position error is within the range determined by the (plus or minus) in position limit set by this parameter, the axis' in-position status bit becomes active.

To set this parameter you must enter the value in machine steps. This value can range between 0 and 65,636. The default value is 65 counts.

VELTRAP

This parameter specifies the maximum instantaneous speed at which an axis may move. A velocity trap occurs if the observed velocity exceeds the amount specified by this parameter. The units for this parameter are machine steps per second and can range from 0 to 65,536,000. The user may enter a zero to disable the velocity trap detection.

FBWINDOW

While processing motion commands, the axis processor integrates both the velocity command and the velocity feedback. This parameter permits you to specify the maximum amount by which these two velocities may differ. A FEEDBACK fault occurs if the difference exceeds the amount specified in this parameter. The units for this parameter are in machine steps and can range from 0 to 1,000,000. A value of zero disables the FEEDBACK fault monitoring.

SAFEZONECW

This parameter allows you to specify the clockwise boundary of the safe zone associated with an axis. The user may use a safe zone to designate a boundary in which the axis can travel, or one in which the axis can not travel. To enable or disable these zones you must properly set the SAFEZONEMODE parameter.

When setting this parameter it is necessary to know that its distance starts at the hardware home position. The units for this parameter are machine steps with a valid range of approximately $\pm 2.1m (2^{31}-1)$. The default for this parameter is zero.

The user may also specify safe zone parameters from within a parts program.



SAFEZONECCW

This parameter allows you to specify the counter-clockwise boundary for the safe zone of an axis. Safe zones are useful for designating an area in which the axis can travel, or one in which the axis can not travel. To enable or disable the specified boundary, you must set the SAFEZONEMODE parameter. The distance specified by this parameter begins at the hardware home position. The units are machine steps with a valid range of approximately $\pm 2.1m (2^{31}-1)$. The default for this parameter is zero.



The user may also specify safe zone parameters from within a parts program.

SAFEZONEMODE

The value set by this parameter determines how the system interprets the SAFEZONECW and SAFEZONECCW parameters. Setting this parameter to zero disables the safe zone, while a one defines an area in which the axis may enter, and a two defines an area in which the axis may not exit.

A safe zone fault occurs each time the associated axis moves into or out of an area that violates an active safe zone.



The user may also specify safe zone parameters from within a parts program.

SIMULATION

To facilitate easy debugging of parts programs, this parameter allows you to place an axis into a simulation mode. While in this mode, the motor's torque remains steady, but no motion occurs. While executing a parts program on a simulated axis, the Axis processor performs all calculations normally, but the torque command never reaches the motor. Instead, the torque serves as the feedback for this axis, effectively creating a system free from velocity error. All other features, such as data acquisition, continue to function normally. Setting this parameter to zero disables the simulation mode, while a one enables it. The default is to disable the simulation mode.

ACCEL

This parameter controls the time needed to accelerate to a new velocity while the ACCELMODE parameter specifies time-based ramping. The units for this parameter are in milliseconds and can range between 0 and 100,000. The default value is for 0 (msec).

Acceleration refers to any increase in velocity.

The user may also specify acceleration mode parameters from within a parts program.

**DECEL**

This parameter controls the time that it takes to decelerate the current velocity to a lesser velocity, while the DECELMODE parameter specifies time-based ramping. The units for this parameter are in milliseconds and can range between 0 and 100,000. The default value is for 0 (msec).

Deceleration refers to any decrease in velocity.

The user may also specify deceleration mode parameters from within a parts program.

**ACCELMODE**

This parameter permits you to select the type of ramping used during the execution of motion commands. This ramping may be time-based (using the ACCEL parameter) or rate-based (using the ACCELRATE parameter). Also, you can configure the ramping to be either linear or sinusoidal (1-cosine). The following chart indicates how to set this parameter. The default for this parameter is zero for a time-based linear ramp.

- 0 - Sinusoidal Ramping - Time Based
- 1 - Linear Ramping - Time Based
- 2 - Sinusoidal Ramping - Rate Based
- 3 - Linear Ramping - Rate Based

The user may also specify acceleration mode parameters from within a parts program.



DECELMODE

This parameter permits you to select the type of ramping used during the deceleration of motion commands. This ramping may be time-based (using the DECEL parameter) or rate-based (using the DECELRATE parameter). Also, you may configure the ramping to be either linear or sinusoidal (1-cosine). The following chart serves as an aid in setting this parameter. The default for this parameter is for a time-based linear ramp.

- 0 - Linear Ramping - Time Based
- 1 - Sinusoidal Ramping - Time Based
- 2 - Linear Ramping - Rate Based
- 3 - Sinusoidal Ramping - Rate Based



The user may also specify deceleration mode parameters from within a parts program.

FEEDRATEMODE

This parameter determines if the axis is subject to feedrate override control. Setting this parameter to zero disables the feedrate override control, while a value of one makes the axis subject to feedrate override. The default value is zero (0).

ACCELRATE

This parameter sets the rate of acceleration, while the ACCELMODE parameter specifies rate-based ramping. The units for this parameter are machine counts per second squared. This parameter has a valid range of 1 to 10,000,000. The default value is 1,000,000 (1 count/msec²).



The user may also specify acceleration parameters from within a parts program.

DECELRATE

This parameter sets the rate of deceleration, while the DECELMODE parameter specifies rate-based ramping. The units for this parameter are machine counts per second squared. This parameter can range from 1 to 10,000,000. The default value is 1,000,000 (1 count/msec²).



The user may also specify deceleration parameters from within a parts program.

HOMESWITCHTOL

To ensure the accuracy of a homing sequence, there must be a minimum distance between the home limit switch and the home marker pulse. Otherwise, the axis processor may miss the first marker pulse, and use the second marker as the home position (before the home offset). The required distance depends on two factors: feedback resolution and home feedrate.

This parameter specifies the minimum distance, in machine steps, that must exist between the home limit and the marker pulse. Failure to maintain this distance causes a HOME_SWITCH_TOLERANCE fault to occur. This parameter has a valid range of 0 to 16,384 machine steps. The default value is zero (0).

FAULT

This parameter allows you to view axis faults and clear those that no longer exist. The parameter is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). When writing to this parameter, the axis processor card attempts to clear all faults corresponding to the bits set in this mask. Bits set to zero have no effect on the system.

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.



FAULTMASK

This parameter determines which faults the system should detect. The parameter is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one enables monitoring of that particular fault type. Conversely, clearing a bit causes the system to ignore the fault.

The user may also edit this parameter using the Bit Mask Control Window.



DISABLEMASK

This parameter determines which faults should cause an axis to be disabled. The disable mask is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one disables the axis on that type of fault (assuming the corresponding bit in the FAULTMASK parameter is set).

The user may also edit this parameter using the Bit Mask Control Window.



INTMASK

This parameter allows you to determine which faults will cause a system interrupt. Upon detection of an interrupt, MAINMENU automatically activates the axis fault and status screen (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). This allows you to determine the type of fault that occurred.

This parameter is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one causes the system to generate an interrupt when that fault occurs (assuming the corresponding bit in the FAULTMASK parameter is set).



The user may also edit this parameter using the Bit Mask Control Window.

AUXMASK

Through this parameter you may designate which faults should turn off the auxiliary output associated with an axis. This parameter is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one turns on the auxiliary output when a particular fault occurs (assuming the corresponding bit in the FAULTMASK parameter is set).



The user may also edit this parameter using the Bit Mask Control Window.

HALTMASK

This parameter controls the faults that cause the axis to halt. If the system must halt motion, the axis will gracefully decelerate to zero velocity based on the time specified in the DECEL parameter. This parameter has no effect on the position error tracking

The value specified for this parameter is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one halts the axis when a particular fault occurs (assuming the corresponding bit in the FAULTMASK parameter is set).



The user may also edit this parameter using the Bit Mask Control Window.

IOLEVEL

This parameter permits you to specify the active state for several of the axis processor I/O lines. The user may configure any of the following lines.

0 - Drive Enable	(Output)
1 - Aux Output Enable	(Output)
2 - CW Limit Switch	(Input)
3 - CCW Limit Switch	(Input)
4 - Home Limit Switch	(Input)
5 - Drive Fault	(Input)

The value specified is a bit mask where only the six least significant bits are valid. These bits correspond with the list above, as well as the six least significant bits of the STATUS parameter (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one implies that the input/output is active high, and a zero means it is active low. The default for this parameter is 63 (03FH), and corresponds to all active high signals.

AUXOFFSET

To understand how this parameter functions, the reader must be familiar with the operation of the synchronized auxiliary output tables on the UNIDEX 631/U600. In brief, each synchronized auxiliary output table entry specifies a master position and a corresponding state for the auxiliary output. When the observed master position becomes greater than or equal to that specified in the table entry, the output gets set to the appropriate state. The only requirement is that the master positions must constantly increase and never repeat.

This parameter refers to an offset applied to the master position of the auxiliary output table associated with an axis. The point at which the table begins and ends is advanced or retarded. The user must be aware of the table's setup before setting the value of this parameter. The units for this parameter are in machine steps, and can be any signed 32-bit numbers. The default value is for zero (0) machine steps.

ABORTMASK

This parameter controls the faults that cause an axis to abort motion. If the system is to abort motion, it disregards the DECEL parameter setting and stops the axis immediately. This also sets the current position error to zero.

The value specified is a bit mask where each bit corresponds to a specific fault (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*). Setting a bit to a one causes the axis to abort motion when that particular fault occurs (assuming the corresponding bit in the FAULTMASK parameter is set).

This parameter can not be set from the Axis Parameter screen. Use the "SETPARAM" command in a program or manual mode to set this parameter.



MASTERPOS

To understand how this parameter works, the reader must be familiar with the operation of synchronized motion through the use of CAM tables on the UNIDEX 631/U600. While operating in this mode, axis motion relates directly to motion on the master axis (the axis designated by you). The basis of this relationship is dependent on the currently active CAM table.

Each CAM table entry contains two position types: a master position and a slave position. As the master axis approaches the positions found within the CAM table, the slave axis moves to the corresponding slave position. Interpolation occurs between the CAM points. The first CAM table entry for the master position must be a zero. Two rules apply to all master positions following the first entry: they must always increase and they must never repeat.

This parameter relates to the current observed position of the master axis. Its primary purpose is to determine the current location within a CAM table, once activated. This parameter cannot be changed when a CAM table is active.

The units for this parameter are machine steps, and are in the range of approximately $\pm 2.1m (2^{31}-1)$. The default value is for zero (0) machine steps.



The user may set the master position of an axis without affecting the POS parameter of the master axis.

CAMOFFSET

To understand how this parameter functions, the reader must be familiar with the operation of the synchronized motion through the CAM tables on the UNIDEX 631/U600. For a brief discussion of this feature, refer to the MASTERPOS parameter (*Refer to the appropriate section of chapter 5 in the UNIDEX 631/U600 User's Manual, EDU153*).

This parameter offsets all master positions in the active CAM table based on a fixed amount. That is, the value of this parameter is added to each master position in the CAM table. Most often you select this parameter to permit execution of the slave profile without forcing the master position to zero (starting point of table).

The units for this parameter are machine steps, and the valid range is $\pm 2.1m (2^{31}-1)$. The default is zero (0) machine steps.

SYNCSPEED

To understand how this parameter works, the reader must be familiar with the operation of the synchronized motion through the CAM tables on the UNIDEX 631/U600. For a brief discussion of this feature, refer to the discussion of the MASTERPOS parameter (*Refer to the UNIDEX 631/U600 User's Manual, EDU153*).

There are two modes in which you can perform CAM table execution. In the first mode, the system assumes that the current slave position is the starting point of the CAM table. Also, the system assumes that all slave position entries are relative to that starting point.

The second mode of CAM table execution does not make that assumption. Instead, it interprets the slave positions found within the CAM table as absolute positions. With synchronization enabled, the system determines the current location within the CAM table based on the current master position. The slave axis then moves to the position that corresponds to the current master position.

This parameter defines the speed at which the slave axis is to move. The units for this parameter are machine steps per second with a valid range of approximately $2.1m(2^{31}-1)$. The default value is 1000.

BASE_SPEED

This parameter, as well as the following two parameters allow the speed torque characteristics of an AC brushless motor to be customized. Normally these parameters are only used with motors having a large back-EMF (K_B) constant. This is done by adjusting each slope of a dual slope curve which determines the torque angle at various motor speeds. The base_speed determines the speed at which the motor will reach a 20° phase advance. The phase_speed parameter determines the speed at which the maximum torque angle will be reached, which is specified by the maximum phase parameter offset.

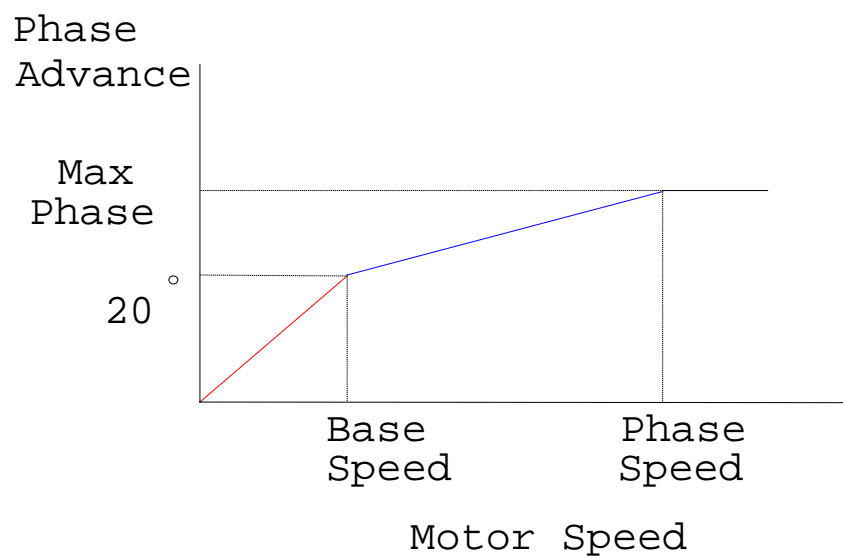


Figure A-1. The Defined Slope of a Velocity Curve at a Specified Angle

As the motor's velocity reaches the base speed, the phase advance reaches 20° . The phase_speed parameter specifies the motor velocity at which the phase advance reaches the max_phase degrees offset.

MAX_PHASE

Refer to the base_speed parameter for a description of this parameter.

PHASE_SPEED

Refer to the base_speed parameter for a description of this parameter.

SOFTLIMITMODE

This parameter sets the active mode for the software limits (defined by the CWEOT and CCWEOT parameters), as well as the safe zones (defined by the SAFEZONECW, SAFEZONECCW, and SAFEZONEMODE parameters). In many systems, the current absolute position of an axis is unknown until after the axis reaches its home position. Therefore, you should not activate a software limit or a safe zone until the system successfully completes the homing process.

However, the mechanics of some systems do not permit execution of a normal homing sequence. Therefore, you must use an alternate method to determine the absolute position. In this case, it may be logical to permit software limits and safe zones to be active at all times.

Setting this parameter to a one (1) causes these features to become active after successfully homing the axis. The default value is zero (0), and causes these features to always be active.

ALPHA

The alpha parameter is used to filter the AFFGAIN parameter, this limits disturbances in the servo loop caused by sudden accel changes that might be generated from a handwheel input with a large scaling factor. The scaling of this parameter is inverse so 65536 is no filtering of the AFFGAIN parameter, and 1 would be maximum filtering.

Range 1-65536

Default 65536

VGAIN

This parameter is used by the servo loop to minimize position error during constant velocity. It is only used when the command to the drive is a velocity command when the Kp and Ki parameters are both zero.

Range 0-8388607

Default 0

ICMDPOLARITY

This parameter is used to invert the phasing of the servo loop. This is done by inverting the current command polarity that is output from the DAC to the drive. On a U631 a positive command should produce clockwise rotation of the motor, on the U600 it should be counterclockwise. This parameter is used only to correct for reversed servo loop phasing, not to reverse the direction of the motor on a properly phased servo loop.

This parameter is typically set to 1 when a Unidex 31 (U631) system is upgraded to a Unidex 600.

Range 0-1

Default 0

EXTR2DSCL (EXtend R/D SCAling)

This parameter is used in conjunction with other parameters to increase the range of the PGAIN parameter, since the minimum value of the PGAIN is 1. This parameter is used to scale the R/D channel and the resolver feedback resolution of the velocity loop. In addition, the parameters for the encoder lines/rev., mm/rev., or in/rev., KP, KI and the PGAIN must all be increased proportionally by the value of this parameter to complete the scaling of the gains.

For example, if the range of the PGAIN parameter is inadequate, increase the EXTR2DSCL parameter to an arbitrary value such as 128. Now, increase the encoder lines/rev parameter within the CONFIG screen proportionally by 128, ie; enter a value 128 times greater. The mm/rev. and in/rev. parameters must also be increased proportionally by 128, ie; enter a value 128 times greater. The KP, KI and PGAIN parameter must also be increased proportionally by 128, ie; enter a value 128 times greater than each of the current values. This will allow a more reasonable range for the PGAIN parameter while maintaining the proper values for the other servo loop parameters.

Range 1-65536

Default 0

HOMEVELMULT

This parameter is used to vary the velocity during the marker search of the home cycle (if machine parameters are set for this type of home cycle). After the home limit (typically this is the CCW limit) is encountered the motor will reverse direction and begin seeking the marker pulse. In this phase of the home cycle as it seeks the marker pulse, this parameter may be used to vary the velocity. The value entered is interpreted as a MFO value, 1 is 1% of the home velocity (specified in the machine parameters) and 100 is 100%.

Range 1-100

Default 100

CAMADVANCE

This parameter is used in the camming mode as a master offset advance that is a function of the masters velocity. The units are in counts that will advance by counts per msec.

Range -100000000 to 100000000

Default 0

VELTIMECONST

This parameter is used by the velocity filter to smooth out the velocity command. The units are in msec. and specify the time to reach the commanded velocity.

Range 0-1000

Default 0

MAXCAMACCEL

This parameter determines the maximum acceleration that may occur in the velocity camming mode (mode 3). The units are counts per msec..

Range 0-65536

Default 0

BRAKEMASK Parameter

On the U600, which has only one brake output; any disabled axis with a non-zero brakemask will engage the brake. The brake will be dis-engaged when ALL axis with their brakemask set are enabled. Also, any of these axis with the brakemask set will cause the brake to be engaged if the axis faults.

On the U631, it is the same as described above for U600 except, each axis may have its own independantly controlled brake because it will be engaged by the axis auxillary (Aux) output.

DACOFFSET Parameter

This parameter is used to null any offset in the command to the drive, which is output by the axis DAC when the command should be at zero. It is entered as a signed number representing the number of counts (+/-) required to bring the output command to zero volts. This value is added to all output DAC values. For example, if the DAC on an axis has a 60mV (.060 volt) offset, -392 would be entered for this parameter. Unidex 600's 16 bit DAC has a maximum output of 10 volts, therefore

$$\begin{aligned} &.060 / \\ & / 10 / \\ & / / (2 ^ 16) = 392 \text{ counts of the DAC} \\ & / \end{aligned}$$

▽ ▽ ▽

APPENDIX B: WARRANTY AND FIELD SERVICE

In This Section:	
• Laser Products.....	2-1
• Return Procedure.....	2-1
• Returned Product Warranty Determination.....	2-1
• Returned Product Non-warranty Determination.....	2-2
• Rush Service.....	2-2
• On-site Warranty Repair	2-2
• On-site Non-warranty Repair	2-2

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 or 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.

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.

Laser Products

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.

Return Procedure

After Aerotech’s examination, warranty or out-of-warranty status will be determined. If upon Aerotech’s examination a warranted 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 not extend the original warranty period.

***Returned Product
Warranty Determination***

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.

Company Address

Aerotech, Inc.
101 Zeta Drive
Pittsburgh, PA 15238-2897
USA

Phone: (412) 963-7470
Fax: (412) 963-7459



APPENDIX C: ERROR CODES**In This Section:**

- DescriptionC-1
- Axis Processor Error CodesC-31

Description

The following section provides a quick reference of MAINMENU.exe error codes listed in alphabetical order. Most display a CNC number, line number and/or program name, where the numbers are shown within the error messages as %ld or %d and the text (strings) are indicated by %s. These are replaced at runtime with the appropriate values.

Allocate CNC%d, Program %d, Lines %ld, ret:%d completion:%d

Memory could not be allocated for a new CNC. Close unnecessary windows or add more memory to your system.

Allocate CNC Queue Failed, stat=%d ret=%d\n

Memory could not be allocated for the CNC queue. Close unnecessary windows or add more memory to your system.

Allocate CNCStack Error, CNC %d, Program %d

Memory could not be allocated for the CNC's user program stack. Close unnecessary windows or add more memory to your system.

Allocate CNCVariable Error, CNC %d, Program %d

Memory could not be allocated for the CNC's user variables. Close unnecessary windows or add more memory to your system.

Allocate Variables CNC %d, Program %d ret:%d completion:%d

Memory could not be allocated for CNC variables. Close unnecessary windows or add more memory to your system.

Allocation error allocating structures for IF statement compilation

Memory could not be allocated for IF structures. Close unnecessary windows or add more memory to your system.

Allocation error allocating structures for WHILE statement compilation

Memory could not be allocated for WHILE structures. Close unnecessary windows or add more memory to your system.

Anchor Block Initialize Failure CNC %d

Internal error possibly due to insufficient memory.

Attempt to read beyond end of line during READFILE

More data was expected. Be sure the file data agrees with the data expected.

Attempted to open too many source files: %s

There were more than 4 nested include files.

Axis not assigned to this CNC

Axis must be assigned to a CNC before they may be used. Check the parameter setup mode to be sure all axis are assigned properly.

B Axis Must be a Rotary Axis, Check Machine Parameters - Line %ld, CNC %d

The axis assigned within the setup parameters as the B axis is not defined as a rotary axis.

B Axis Not Selected, Check CNC Parameters - Line %ld, CNC %d, Program %d

No axis has been designated as the B axis within the setup parameters.

B Axis User Units are Zero, Check Axis Parameters - Line %ld, CNC %d

The conversion factor is zero for the axis assigned as the B axis within the setup parameters.

B Axis X Plane Axis Not Selected, Check CNC Parameters - Line %ld, CNC %d

No axis has been assigned to the X plane for the B axis within the setup parameters.

B Axis Y Plane Axis Not Selected, Check CNC Parameters - Line %ld, CNC %d

No axis has been designated as the Y plane of the B axis.

Cannot use HardCoded Axis Names When Softnames are Selected

Select Hardcoded axis programming mode, or specify the axis by their Softnames.

CNC %d, Program %d ret:%d completion:%d - Program Line %ld

An error occurred while transferring a CNC command block to the axis card.

CNC%d Execute Immediate Error, ret:%d compl:%d

An error occurred while executing the CNC command line.

CNC: %d, Specified Axis Mask: %lx, Actual Axis Mask: %lx

The axis were re-assigned in the parameter setup screen - the system must be initialized again.

Connection with Host: %s Failed: %s

Unable to communicate with specified host. Check the host address, host power, and U31/host connection to Ethernet.

Constant memory allocation error

Memory could not be allocated for constant. Close unnecessary windows or add more memory to your system.

Constant name too long

Constants may not be longer than 20 characters, less than 2 characters and the first 3 characters must be alphabetic. All others may be alphanumeric.

Constant previously defined

This constant was previously defined. Assign a new name or remove previous definition.

Could Not Obtain Display Handle, CNC: %d, Program: %d

Internal error - unresolvable by user.

Create Run Window Failure CNC %d

The run window could not be created possibly due to insufficient memory.

Cutter X undefined

No axis has been assigned to the X Plane for ICRC control.

Cutter Y undefined

No axis has been assigned to the Y Plane for ICRC control.

Define stack overflow

You have exceeded the limits of UNIDEX 31's user stack.

DMRLoadCNCLine on END, Return=%d Completion=%d

An error occurred loading the command block to the axis card.

DMRLoadCNCLine return=%d completion=%d

An error occurred while transferring a CNC command block to the axis card.

DMRLoadCNCQueue return=%d completion=%d

An error occurred while transferring a CNC command block into the Ethernet queue.

DMRSetCNCBlockDeleteMode CNC %d return=%d completion=%d

An error occurred while disabling block delete on the CNC.

DMRSetCNCFeedrate CNC %d return=%d completion=%d

An error occurred while setting the MFO on the CNC.

DMRSetCNCOptionalStopMode CNC %d return=%d completion=%d

An error occurred while clearing optional stop on the CNC.

DMRSetCNCSingleStepMode CNC %d return=%d completion=%d

An error occurred while setting the CNC into single step mode.

DMRSetCNCSpindleFeedrate CNC %d return=%d completion=%d

An error occurred while setting the MSO on the CNC.

DO without WHILE Statement

You have specified a DO statement without a required WHILE statement.

DoCNCMCode return=%d completion=%d

An error occurred decoding a before/after M code.

DoCNCNonMove return=%d completion=%d

An error occurred decoding the G36/G37, G54, G55, G60, G61, G62, G63/G64, G65, G66, G67/G68, G70/G71, G90/G91, G93/G94/G95, G96/G97, G98/G99 or G110/G111 command.

ELSE without an IF statement

You have specified an ELSE statement without a required IF statement.

ENDIF without a corresponding IF

You have specified an ENDIF statement without a required IF statement.

ENDRPT without a RPT statement

You have specified an ENDRPT statement without a required RPT statement.

ENDWHILE without a corresponding WHILE

You have specified an ENDWHILE statement without a required WHILE statement.

Entry point stack overflow

You have exceeded the limits of UNIDEX 600/631's user stack.

Error Allocating string pointers, CNC %d, Program %d

Memory could not be allocated for the CNC's string pointers. Close unnecessary windows or add more memory to your system.

Error Allocating string space, CNC %d, Program %d

Memory could not be allocated for the user file to be opened. Close unnecessary windows or add more memory to your system.

Error receiving Ethernet Packet: %s

An error has occurred while receiving an Ethernet packet from the host.

Exceeded If nesting level

If statements may only be nested 20 levels deep.

Exceeded maximum RPT nesting level

Repeat loops may only be nested 20 levels deep.

Exceeded Maximum Variables

Local and global variables are limited to 1000 each.

Exceeded WHILE nesting level

While loops may only be nested 20 levels deep.

Execute Immediate CNC %d Command Timeout

The command failed to execute after 100 attempts.

Failed to open source file

The specified source file could not be found or it was not located in the default location and no alternate file path was specified.

Failure reading FILEREAD data on CNC %d

An error occurred while reading data from the file. Verify the datafile and expected data agree.

Failure to parse READFILE data

Unable to properly read the parameters to the READFILE command. Verify the READFILE command line.

Failure to parse WRITEFILE date

Unable to properly read the parameters to the writefile command. Verify the writefile command line.

Failure writing FILEWRITE data on CNC %d

An error occurred while writing the data to the file.

Feedrate Must be Between %lf and %lf

The feedrate is not within the range determined by the min/max vector feedrate parameters.

Feedrate Must be Between %lf and 0.0

The feedrate is not less than the maximum vector feedrate and greater 0.

Free CNC %d, Program %d ret:%d completion:%d

The memory allocated for a program run could not be freed.

FreeVariables CNC %d, Program %d ret:%d completion:%d

The memory allocated for the CNC variables could not be freed.

"G33 Error: Linear Feedrate Cannot be a Variable, Make Linear Feedrate a Literal - Line %ld, CNC %d, Program %d"

The feedrate is not a literal.

G33 Parallel Axis Does Not Match, Check CNC Parameters - Line %ld, CNC %d, Program %d

The ThreadX axis defined in the setup parameters is not assigned to this CNC.

G33 Parallel Axis Not Linear, Check CNC Parameters - Line %ld, CNC %d, Program %d

The axis assigned within the setup parameters as the parallel axis is not defined as a linear axis.

G33 Parallel Axis Not Specified, Check CNC Parameters - Line %ld, CNC %d, Program %d

No parallel axis was specified in the setup parameters.

G33 Parallel Max Speed is Zero, Check CNC Parameters - Line %ld, CNC %d, Program %d

The maximum speed for the ThreadX axis is zero.

G33 Parallel User Units are Zero, Check Machine Parameters - Line %ld, CNC %d, Program %d

The conversion factor is zero for the axis assigned as the parallel axis within the setup parameters.

G33 Perpendicular Axis Not Specified, Check CNC Parameters - Line %ld, CNC %d, Program %d

No perpendicular axis was specified in the setup parameters.

G33 Perpendicular Max Speed is Zero, Check CNC Parameters - Line %ld, CNC %d, Program %d

The maximum speed for the ThreadY axis is zero.

G33 Perpendicular User Units are Zero, Check Machine Parameters - Line %ld, CNC %d, Program %d

The conversion factor is zero for the axis assigned as the perpendicular axis within the setup parameters.

Get CNC %d Status Error, ret:%d, Completion:%d

An error occurred while reading the CNC status from the axis card.

GETPARAM Parameter not readable

The parameter specified to read the value of in the GETPARAM command is not recognized.

Global Variable Designation Out of Range

Global variables are limited to 1000.

Halt CNC Failure on CNC %d.

An error occurred while trying to stop the current CNC program.

Halt Error CNC %d, Program %d ret:%d completion:%d

The current CNC program could not be halted.

Identifier More than 8 Characters

Identifiers are limited to 8 characters.

If Pointer Compiler Error

Internal error

Illegal CLLS Syntax

Invalid syntax, verify command syntax.

Illegal DENT Syntax

Invalid syntax, verify command syntax.

Illegal DFS Syntax

Invalid syntax, verify command syntax.

Illegal CLS Syntax

Invalid syntax, verify command syntax.

Illegal JUMP Syntax

Invalid syntax, verify command syntax.

Invalid HOME Syntax

Invalid syntax, verify command syntax.

Illegal RPT Syntax

Invalid syntax, verify command syntax.

Illegal DFLS Syntax

Invalid syntax, verify command syntax.

Illegal Feedrate

The feedrate that you have specified is not valid.

Illegal filename length

Filenames may be no greater than 8 characters followed by three characters in the format 'XXXXXXXX.XXX'.

Illegal filename specified

The filename that you have specified is not valid.

Illegal G33 Syntax

The format that you have used for the G33 command is incorrect.

Illegal G36 Mode Specified

The valid modes for the G36 command are P1, and P2.

Illegal G36 Syntax, Missing Positive Safezone Parameter

You did not specify the positive limit for an axis in the safezone command.

Illegal JUMP target

The label specified for the jump is not valid.

Illegal ONERRGOTO syntax

You must specify a program label to jump to on an error condition.

Illegal ONERRGOTO target

The label you specified to jump to is not valid.

Illegal POP syntax

The POP command may have one variable specified to hold the value removed from the user stack.

Illegal Probe Offset Masks on CNC %d

There are no axes defined for the touch probe.

Illegal PUSH syntax

You did not specify the PUSH command properly - PUSH < data > .

Illegal Spindle Feedrate

The spindle feedrate you have specified is not valid.

Illegal T Code syntax

You did not specify a valid T-code.

Illegal TWord Set %d Data in CNC %d

Error in tool file.

Illegal variable redefined

The variable can not be redefined.

Incompatible assignment

The mathematical assignment is not possible.

Incomplete Data Command

You did not specify all parameters to the data command.

Initialize Axis Parameter String Failure on CNC %d.

An error occurred while retrieving the axis parameter names from the axis card.

Invalid Array Designation

The array specified is not valid.

Invalid assignment target

The specified assignment can not be made.

Invalid assignment

This assignment can not be made.

Invalid CNCNUM syntax

The valid range for the Cncnum command is 0 -3.

Invalid Constant designation

The specified constant is not valid.

Invalid cutter mask definition

Less than two axes have been assigned for ICRC control.

Invalid DISPLAY return specifier

The variable specified to return the value from the user is not valid.

Invalid EXECUTE return code variable specification

The variable specified to hold the return code from the EXE program to execute is not valid.

Invalid EXECUTE syntax

The syntax for the execute command is - (EXECUTE \path\filename.exe=var1). The first character following the execute command (path or filename) must be an alphabetic character so that it is recognized as a valid path or filename.

Invalid expression

The indicated expression is not allowed.

Invalid Extended Command

The command specified within the parenthesis is not a valid extended command.

Invalid Feedrate Type, Non-Motion G Code - %d - Line %ld, CNC %d, Program %d

The feedrate was not a variable (math_pointer) or a literal.

Invalid field

The field is not valid.

Invalid FILECLOSE file pointer returned from CNC

There are too many user files open or an internal error has occurred corrupting the file pointer.

Invalid FILECLOSE syntax

The FILECLOSE command was not specified properly.

Invalid FILEOPEN syntax

The FILEOPEN command was not specified properly.

Invalid FILEREAD file pointer returned from CNC

There are too many user files open or an internal error has occurred corrupting the file pointer.

Invalid FILEREAD syntax

The FILEREAD command was not specified properly.

Invalid FILERESET syntax

The FILERESET command was not specified properly.

Invalid FILERESSET file pointer returned from CNC

There are too many user files open or an internal error has occurred corrupting the file pointer.

Invalid FILEWRITE file pointer returned from CNC

There are too many user files open or an internal error has occurred corrupting the file pointer.

Invalid FILEWRITE syntax

The FILEWRITE command was not specified properly.

Invalid First Token

The first parameter to the command is invalid.

Invalid E Code

Its parameter was neither numeric or a variable.

Invalid F Code

Its parameter was neither numeric or a variable.

Invalid G Code

There was no G-code number following the “G”.

Invalid I Code

Its parameter was neither numeric or a variable.

Invalid J Code

Its parameter was neither numeric or a variable.

Invalid K Code

Its parameter was neither numeric or a variable.

Invalid M Code

There was no M-code number following the “M”.

Invalid N Code

There was no N-code number following the “N”.

Invalid O Code

Its parameter was neither numeric or a variable.

Invalid P Code

Its parameter was neither numeric or a variable.

Invalid Q Code

Its parameter was neither numeric or a variable.

Invalid R Code

Its parameter was neither numeric or a variable.

Invalid S Code

The parameter was neither numeric or a variable.

Invalid G84 (ROTATE) axis mask

The axis specified for the command are not permitted. Be sure they are assigned to this CNC.

Invalid G84 (ROTATE) syntax

The syntax is - G84 axis1, axis2, angle.

Invalid G96 Code - %d, Check ThreadY Axis Resoluton in Machine Parameters - Line %ld, CNC %d, Program %d

ThreadY axis conversion factor equals zero.

Invalid G96 Code, Check ThreadY Axis Selection in Machine Parameters - Line %ld, CNC %d, Program %d

No ThreadY axis has been defined.

Invalid G96 Code, No Spindle Selected in CNC Parameters - Line %ld, CNC %d, Program %d

No spindle axis selected in CNC parameters.

Invalid GETPARM axis mask

The axis whose parameter was specified to be read is not valid. Be sure that it is assigned to this CNC.

Invalid GETPARM Variable Assignment

The variable specified to hold the value of the specified axis parameter is not valid.

Invalid Global or Local Specification

The local or global variable specified is outside the range of variables defined within the variable allocation group box of the parameter setup mode.

Invalid Hardcoded Axis Code

The hardcoded axis specified is not valid. Check parameter setup mode and be sure the axis is assigned to this CNC.

Invalid identifier usage

The identifier shown can not be used in this context.

Invalid index type

The index is of the wrong type.

Invalid Manual Mode Command

This command is not valid in the manual mode.

Invalid Monitor axis designation

You have not specified a valid axis to monitor.

Invalid Monitor Parameter designation

You have not specified a valid parameter to monitor.

Invalid Monitor trigger level designation

You have not specified the trigger level properly.

Invalid Non-Motion G Code - %d - Line %ld, CNC %d, Program %d

The G code number is not valid.

Invalid number

The number is out of range.

Invalid Parameter name

The specified parameter name is not valid.

Invalid Parameter number

The specified parameter number is not valid.

Invalid Plane Selection, Check the Plane Setup Menu for current XYZ Plane Selection

The current axis plane does not have three axes assigned to it or those axes are not assigned to this CNC.

Invalid Probe Syntax

The syntax is the PROBE keyword, C is the channel number, L is the active level, and an array variable.

Invalid PSOC Syntax

The syntax is incorrect.

Invalid PSOD Syntax

The syntax is incorrect.

Invalid PSOF Syntax

The syntax is incorrect.

Invalid PSOM Syntax

The syntax is incorrect.

Invalid PSOP Syntax

The syntax is incorrect.

Invalid PSOR Syntax

The syntax is incorrect.

Invalid PSOT Syntax

The syntax is incorrect.

Invalid PSOC 4 Configuration, last two numbers must add up to 3

The I/O bits were not configured properly. See manual for valid options.

Invalid PSOC Input

The specified input is not valid. Is it configured as an input by the PSOC,4 command.

Invalid PSOC Level

The valid levels are 0, 1 or x (don't care).

Invalid PSOC type, must be 0,1,2, or 3

These are the only valid forms of the PSOC command

Invalid PSOD type, must be 0,1, or 2

These are the only valid forms of the PSOD command.

Invalid PSOF type, must be 0,1,2,3,4 or 5

These are the only valid forms of the PSOF command.

Invalid PSOM type, must be 0

The PSOM,1 command is not supported on UNIDEX 600/631.

Invalid PSOP type, must be 0,1,2,3,4 or 5

These are the only valid PSOP commands.

Invalid PSOR type, must be 0,1 or 3

These are the only valid PSOR commands.

Invalid PSOT type, must be 0,1,2,3,4, or 6

These are the only valid PSOT commands.

Invalid Radius specifier on G43

The tool diameter may be set to zero.

Invalid Radius Specifier on G96

The distance specified from the tool tip to the center of the spindle axis is invalid.

Invalid relational operator

The indicated operator is not a valid relational operator.

Invalid SETPARAM axis mask

The specified axis mask is not valid, or the axis are not assigned to this CNC.

Invalid soft axis name

This soft axis name has not been assigned or it is assigned to an axis on another CNC

Invalid starting statement

The starting statement is not valid.

Invalid statement

This statement is not valid.

Invalid subrange type

This subrange is not valid.

Invalid type

This type is not valid.

Invalid DISPLAY syntax

The syntax is - DISPLAY "message" [= var [/BUTTON1]].

Local Variable Designation Out of Range

Local variables are limited to a maximum of 1000.

M Code data not found in MCODE.INI

The MCODE.INI file is blank or contains no valid M code initialization statements.

M Code Get Binary Input Data Failure

M-code I/O failure.

M Code Get Binary Output Data Failure

M-code I/O failure.

M Code Get Register Input Data Failure

M-code I/O failure.

M Code Get Register Output Data Failure

M-code I/O failure.

M50 No Spindle Defined - Line %ld, CNC %d, Program %d

No spindle axis has been defined.

Math stack overflow, reduce the number of operations on this line

You have exceeded the limit of UNIDEX 600/631's math operations.

Message Queue Creation Failure CNC %d

Internal error possibly due to insufficient memory.

Min limit greater than max limit

Set the maximum limit greater than the minimum limit.

Missing DO

You omitted a DO statement.

Missing END

You omitted an END statement.

Missing ENDIF Statement

An ENDIF was found without a corresponding IF statement.

Missing ENDRPT statement

An ENDRPT was found without a corresponding RPT statement.

Missing ENDWHILE Statement

An ENDWHILE was found without a corresponding WHILE statement.

Missing equal

You did not equate the expression to anything.

Missing File Pointer variable

No variable was specified for the file pointer.

Missing identifier

You did not specify an identifier.

Missing Left Bracket

A left bracket was found without a terminating right bracket.

Missing Left Parenthesis

A right parenthesis was found without an opening left parenthesis.

Missing Program

You did not specify a program name.

Missing Right Bracket

A right bracket was found without an opening left bracket.

Missing Right Parenthesis

A left parenthesis was found without a terminating right parenthesis.

Missing RPT Statement

An ENDRPT statement was not preceded by a RPT statement.

Missing THEN

An IF statement was not followed by a THEN statement.

Missing variable

You did not specify a variable.

MR Mcode associate failure

An error occurred while reading the M-code data.

Nesting too deep

Nesting may not exceed twenty levels.

No error

No error has occurred.

No Linear Feedrate, CNC %d, Program %d, Line %ld

The linear feedrate equals zero.

No Rotary Feedrate, CNC %d, Program %d, Line %ld

The rotary feedrate equals zero.

No Spindle Defined - Line %ld, CNC %d, Program %d

No spindle axis has been defined in the setup parameters.

No Spindle Defined - Line %ld, CNC %d, Program %d

No spindle axis has been defined.

No variables allowed in this statement.

You may not use variables in this statement.

"Number out of range"

The number is out of range.

Open include part program file failed

The included parts program was unable to be opened. Verify filename, path, and physical location.

Operation on unopened user file

The specified operation can not be performed on the user file because it has not been opened.

Program Translation Error CNC %d, Program %d, Line: %ld

This error occurred on the line number, compiler pass and program line as indicated.

READFILE statement requested more than 30 variables

The READFILE statement is limited to 30 variables.

READFILE write variable error

An error occurred while parsing the command line and counting the number of variables to be read.

Redefined identifier

You have redefined the indicated identifier. This is not allowed.

Register Class Failure CNC %d

An error occurred while attempting to register a new window class, possibly due to insufficient memory.

Resolve MR code failure

An error occurred while reading the M code data. Check the syntax for the M code.

Restore Token Error

A token could not be placed back into the command queue, probably due to one having already been restored.

Return: %d, CompletionCode: %d

An error occurred while retrieving the data for the file requested to be opened, closed, read, written or reset by the axis processor or an error reading or setting the T word data from the tool file.

Rotary feedrate must be a literal on M05,M19 - Line %ld, CNC %d, Program %d

Rotary feedrate is not a literal.

Rotary Feedrate on B Axis Move must be a Literal - Line %ld, CNC %d, Program %d

The rotary feedrate on B axis move must be a literal.

Rotary Feedrate on B Axis Move must be between 0.0 and %lf, Check E Feedrate - Line %ld, CNC %d, Program %d

The feedrate is not greater than zero and less than or equal to the maximum speed assigned to the B axis.

Rotary Feedrate on B Axis Move must be between 0.0 and %lf, Check Axis Parameters - Line %ld, CNC %d, Program %d

Rotary feedrate is not greater than zero and less than the maximum B axis speed.

Rotary Maximum Feedrate on B Axis Move is Zero, Check Axis Parameters - Line %ld, CNC %d, Program %d

Maximum feedrate for the B axis is zero.

RPT memory allocation error

Internal error.

RPT pointer error

Internal error.

Run Thread Activation Failure on CNC %d

The background thread for the run mode could not be started.

Run Thread Failed to Load on CNC %d.

The background process required for the run mode could not be started. This may be due to insufficient system resources; close unnecessary windows or add more memory to your system.

Run Time Error CNC %d, Program %d

The indicated error occurred while running the user motion program.

Server Queue Creation Failure CNC %d, return: %d

Internal error possibly due to insufficient memory.

Set Run Mode Failure on CNC %d.

An error occurred while setting the auto run mode

SETPARM Parameter not writeable

You have attempted to write to a read-only parameter.

Soft axis name specified when hardcoded axis mode is enabled

Select Softcoded axis programming mode, or specify the axis by their Hard coded axis names.

Spindle Feedrate Must be Between %lf and %lf

The spindle feedrate is not within the range determined by the min/max spindle feedrate parameters.

Spindle is Not Defined - Line %ld, CNC %d, Program %d

The spindle axis is invalid.

Spindle Speed = 0.0 - Line %ld, CNC %d, Program %d

The spindle speed is zero.

Spindle Speed must be a Literal on G33 - Line %ld, CNC %d, Program %d

The spindle speed specified was not a literal.

Spindle Surface Feedrate Must be Between Greater than Zero

The spindle surface feedrate is less than zero

Stack Overflow on a DFSL Statement

The user stack has exceeded its limit while defining a library subroutine due to a programming error or a large amount of stack usage.

Stack Overflow

The user stack has exceeded its limit due to a programming error such as a recursive subroutine CALL, a GOTO RETURN from a subroutine, etc. or exceeding the nesting limits of UNIDEX 600/631.

Syntax error

The format for the command is incorrect.

T Word %d Info from %s

An error occurred reading an entry from the tool file.

The number of values read do not match the variables requested in a READFILE statement

There were more/less variables in the data file than specified in the READFILE statement.

THEN without a corresponding IF

A THEN statement was encountered without a corresponding IF statement.

Too many After M codes

User specified too many M-codes to execute after the program block.

Too many Before M codes

User specified too many M-codes to execute before the program block.

Too many digits

Too many digits were specified.

Too many subscripts

You have exceeded the number of subscripts allowed by UNIDEX 600/631.

Too Many Syntax errors

You have too many programming errors.

Too many user files open

You may not open more than 16 user files at a time.

Tool File name invalid, check CNC Parameters

The tool file name specified in the parameter setup menu is invalid. Verify the name and path.

Tool X Assignment Missing, CNC %d, Program %d, Line %ld

No axis is assigned to the X tool.

ToolX Not Selected, Check CNC Parameters - Line %ld, CNC %d, Program %d

No axis has been defined for the X plane axis.

Unable to Allocate Local Variable Memory Space on CNC %d

Out of memory - add more memory to your system.

Unable to Allocate Memory Space for Global Variables

Out of memory - add more memory to your system.

Unable to Allocate Memory Space on CNC %d, Program, %d, Variable: %d\n

Out of memory - add more memory to your system.

Unable to calculate speed for move

The actual speed after calculation was zero. You have a programming error, your axis are not configured properly or the proper dominant feedrate mode is not active.

Unable to calculate vector speed, no axis associated with this CNC

No axis have been assigned to this CNC within the parameter setup menu.

Unable to Display Message on Display Menu CNC %d

An error occurred displaying the message on the customer display window.

Unable to get FILECLOSE data from CNC

An error occurred while retrieving the file close data from the axis processor board.

Unable to Get Filename for Recording CDW Messages CNC %d

An error occurred retrieving the filename for the Customer Display Window messages.

Unable to Get Filename for Recording Strip Data CNC %d

An error occurred retrieving the data for the strip chart recorder.

Unable to get FILEOPEN data from CNC

An error occurred retrieving the data for the FILEOPEN command.

Unable to get FILEREAD data from CNC

An error occurred retrieving the data for the FILEREAD command

Unable to get FILERESET data from CNC

An error occurred retrieving the data for the FILERESET command.

Unable to get FILEWRITE data from CNC

An error occurred retrieving the data for the FILEWRITE command.

Unable to get TWord data from CNC

An error occurred retrieving the Tool data from the axis processor board.

Unable to Obtain Socket Descriptor for Host: %s

A connection to the host was unable to be established over the Ethernet connection. Verify host address, parameter setup and physical connections.

Unable to Open %s on CNC %d

The file requested to be opened by the axis processor could not be opened.

Unable to open Crunch File, CNC %d, Program %d

The temporary intermediate file '\U31\PROGRAMS\INTMFx.TMP' used by the compiler could not be created. The hard drive may be full or a file by that name may already exist.

Unable to open Crunched File, CNC %d, Program %d

The temporary intermediate file '\U31\PROGRAMS\INTMFx.TMP' used by the compiler could not be created. It may have been inadvertently deleted by another process.

Unable to Open File for Recording CDW Messages CNC %d

An error occurred opening the file for recording the Customer Display Window messages.

Unable to open file: %s

The specified source or include file was not found in the specified directory, \U31\Programs*. *' by default.

Unable to Open Get CDW Message Data From CNC %d

An error occurred retrieving the customer display window data.

Unable to Open Intermediate File

The temporary intermediate file '\U31\PROGRAMS\INTMFx.TMP' used by the compiler could not be created. The hard drive may be full or a file by that name may already exist.

Unable to Open Message Display Menu CNC %d

The customer display window could not be opened, possibly due to not enough memory.

Unable to open user file

The specified user file could not be opened. Verify the path and filename are correct.

Unable to read READFILE data from user file

This is due to a bad filename, too many variables were specified to read, a bad variable was specified or the end of file was encountered prematurely.

Unable to read Tool data

This is due to a bad tool filename or an empty or invalid tool file.

Unable to Resolve Host Address: %s

No host was found on the network by the name specified in the setup parameters

Unable to resolve variable in WRITEFILE statement

Unknown variable, verify spelling and variable definition.

Unable to Set Probe Offset Data in CNC %d, ret: %d, CompletionCode: %d

Unable to set T word data.

Unable to Set T Word %d data

The X axis cutter is undefined.

Unable to Set T Word %d data, Cutter Y Axis Undefined, CNC %d

The Y axis cutter is undefined.

Unable to start the run background thread on CNC %d.

The background process required for the run mode could not be started. This may be due to insufficient system resources; close unnecessary windows or add more memory to your system.

Unable to write WRITEFILE data to disk

The file is not open, the command parameters could not be parsed or an unknown variable was found.

Undefined identifier

This identifier has not been defined.

Unexpected end of file

The end of the file was reached before all data was found.

Unexpected Right Bracket

A right bracket has been found without a preceding left bracket.

Unexpected token

A command or parameter was encountered when one was not expected. Verify the command syntax.

Unimplemented feature

This feature has not been implemented.

Unimplemented G Code

This G code has no implementation.

User Units on Fixture Offset 1 are Zero, Check Axis Parameters - Line %ld, CNC %d, Program %d

The conversion factor for fixture offset 1 is zero, check axis parameters.

User Units on Fixture Offset 2 are Zero, Check Axis Parameters - Line %ld, CNC %d, Program %d

The conversion factor for fixture offset 2 is zero, check axis parameters.

User Units on SafeZone Axis are Zero, Check Axis Parameters - Line %ld, CNC %d, Program %d

The conversion factor for the axis defined as the safezone axis is zero.

User Units on ToolX are Zero, Check Axis Parameters - Line %ld, CNC %d, Program %d

The conversion factor for the X plane axis is zero.

Waiting on a Connection with Host: %s

The connection to the host has not yet been established.

While Pointer Compiler Error

Internal error.

Wrong number of parameters

An incorrect number of parameters were specified for the command.

Axis Processor Error Codes

The following is a listing of axis parameter error codes for the axis processor board. When an error occurs, the operator will see a CNC fault condition displayed on the screen with a CNC number indicated by it.

API Probe overtravel limit

The analog values obtained by the API probe have exceeded the values set within the API probe configuration screen. Verify the values are within range and adjust the values within the API probe configuration screen as required.

Axis Disabled

The specified axis is disabled.

Bad Cutter Comp Mask

The same axis has been specified for the cutter axis and one of the cutter compensation axis, both cutter compensation axis are designated as the same axis or one of the axis are not valid for this CNC. Redefine the ICRC axis in the setup mode or with the G44 command.

Bad Cutter Comp Radius

The cutter radius specified is less than zero. Correct this in the parameter screen or specify it correctly with the G43 command.

Bad Safe Zone Mode

You have specified a safe zone mode other than mode 1 or 2.

Bad Thread Master Length

You have specified a zero length for the threading operation.

Can't happen

No error has occurred.

CLS Failed - Stack Full

The specified subroutine cannot be called because the user stack which stores the return line number is full. You have attempted to make a subroutine call with 10 subroutine calls (return values) already on the user stack or you have a programming error.

CNC Axis Fault

An axis has encountered a condition defined within its fault mask as an error. Clear or reset the error and continue.

CNC Probe Data

The probe channel defined is greater than 511 or an invalid variable was specified to hold the data from the touch probe.

Continuous Downloading Queue Full

You have filled the input queue. Lower your command transmission rate or wait for previous commands to execute before sending more.

Cutter Comp Type 1

Axis cutter offsets were active with ICRC disabled during a G2/G3 command.

Cutter Compensation Was Not Terminated

Your program ended while ICRC was active. You must deactivate ICRC before the end of your program with a G40 command.

Emergency Stop

The emergency stop input has been activated.

Feedback Input

While executing an M-code, its corresponding fault feedback input was activated by the external device indicating a fault.

G2 Radius Error

The current position and the end point of the arc have a radius variance of $> .01$. Recalculate either the end point or the center point of the circle.

G3 Radius Error

The current position and the end point of the arc have a radius variance of $> .01$. Recalculate either the end point or the center point of the circle.

Get Parm failure

An error occurred while reading an axis parameter. This may occur when UNIDEX 600/631 must read a system parameter required for the execution of a motion command.

Internal CNC Error Call Aerotech (412) 963-7470

Please provide information regarding the conditions causing this error. This includes your user program and any I/O condition that may have generated this error.

Internal Feedrate Fault

An internal error has occurred while calculating the vectorial feedrate for the axis in motor units.

Internal Profiling Error

An internal error has occurred while setting the move velocities.

Invalid Accel Rate

The acceleration or deceleration rate specified in the G65 / G66 command was less than or equal to zero.

Invalid Accel Time

The acceleration time specified in the G60 command was less than or equal to zero.

Invalid Data Type

The move, speed or other data specified was neither a variable nor a literal.

Invalid Decel Time

The deceleration time specified in the G61 command was less than or equal to zero.

Invalid E Feedrate

The E feedrate specified was less than or equal to zero.

Invalid F Feedrate

The F feedrate specified was less than or equal to zero.

Invalid Index Type

The index that you have specified is not valid.

Invalid Math Operation

The operation that you have specified is neither logical, trigonometric, or mathematical.

Invalid Motion Mask

The specified axis are not assigned to this CNC or are executing asynchronous motion at this time. Verify the axis assignment within the parameter setup screen.

Invalid Pointer Type

You have specified an invalid variable.

Invalid Profile Time

The profile time that you have specified in the G62 command must be greater than .001 seconds and less than or equal to .1 seconds.

Invalid PSO Type

The PSOx command that you have specified is not valid; PSOC, PSOD, PSOF, PSOM, PSOP, PSOR and PSOT commands are valid.

Invalid PSOC Type

The only valid PSOC commands are PSOC,0 thru PSOC,4.

Invalid PSOD Type

The only valid PSOD commands are PSOD,0 thru PSOD,2.

Invalid PSOF Type

The only valid PSOF commands are PSOF,0 thru PSOF,5.

Invalid PSOM Type

The only valid PSOM commands are PSOM,0 thru PSOM,1.

Invalid PSOP Type

The only valid PSOP commands are PSOP,0 thru PSOP,5.

Invalid PSOR Type

The only valid PSOR commands are PSOR,0, PSOR,1, and PSOR,3.

Invalid PSOT Type

The only valid PSOT commands are PSOT,0 thru PSOT,2, PSOT,4, PSOT,6, and PSOT,8.

Invalid Queue Programmed Type

You have specified an invalid program command type.

Invalid Return From Subroutine

You have specified a return from subroutine without a call to a subroutine.

Invalid Spindle Axis

The specified axis may not be used as the spindle axis by the M19 command. It is either an invalid axis or not active.

Invalid Stack Type

While evaluating an expression, an invalid mathematical operation was found or you attempted to PUSH a value onto the user stack that was neither a variable or a literal.

Invalid Threading Command No Spindle Specified

There is no spindle axis active. Either it has not been assigned or an invalid axis has been assigned.

IO Failure

An error has occurred while executing an M code feedback or an invalid probe channel was specified.

Look Ahead Failure

You have attempted to execute a G2/G3 arc after a G41/G42 ICRC enable command with axis other than those defined as the ICRC axis. Verify the axis commanded to move and your ICRC axis assignments within the parameter setup screens.

Mask Not Associated

An invalid axis was specified or the axis is not enabled.

Mask Not Specified

No axis was specified for the G56 command or assigned as the spindle axis.

Normalcy Speed exceeded

The maximum speed of the tool, while maintaining normalcy to the cutting surface has been exceeded. The speed at which this error message is generated is determined by the value entered in the B Axis initialization dialog box. You must lower the programmed vectorial feedrate.

Pendant Data

An invalid command was received from the teach pendant. Be sure that it is properly installed and configured.

Program terminated with Cutter Compensation Active

You tried to define the cutter compensation radius or axis while cutter compensation was active. Deactivate cutter compensation with the G40 command first.

PSO Configuration

You have not properly configured the PSO card with the PSOC,4 command. There are three groups of 8 I/O signals. Each group may be an input or an output. You specify the number of groups that are to be configured as inputs and the rest must be configured as outputs. The two parameters must add up to three, i.e. all three groups must be assigned as inputs or outputs.

PSO invalid PSO Bit

You must specify an I/O bit in the range of 0 thru 23 and the bit must have been configured as an input.

PSO Length

You have specified more data than the PSO card is capable of storing. It is limited to 5000 variables.

PSO Mask

You have specified an invalid axis as a parameter to the PSO card. The specified axis must be assigned to this CNC.

PSO not implemented

The PSOM,1 command has not been implemented by UNIDEX 600/631.

PSO Timeout

The PSO board is not responding to the axis processor. Be sure that it is properly installed, if so, press the reset button on the PSO card to reset it.

Set Parm failure

An error occurred while modifying an axis parameter.

Spindle Resolution is Zero

The axis assigned as the spindle axis has an invalid scale factor. Check the spindle axis assignment and its configuration.

Stack Empty

You have attempted to remove (POP) a data value from the user program stack when the stack contains no data. Verify your program logic. You must PUSH data onto the stack before it may be removed (POPed) from the stack.

Stack Full

The user program stack is full. You have filled the stack with data or you have called too many subroutines, defined too many repeat loops or have a programming error.

The normalcy axis is not a rotary axis

The B Axis must be defined as a rotary axis within the B Axis configuration initialization group box.

Unknown CNC Type

This command is not recognized as any valid UNIDEX 600/631 command.

Unknown G Code Type

The G code specified has no defined function within UNIDEX 600/631.

Unknown M Code Type

The specified M code is not recognized as a binary input/output or a register input/output command.

▽ ▽ ▽

SYMBOLS

-, 4-1
 \$, Special Symbol, 3-5
 (, 1-1
), 4-12
), 1-2
 *, 4-10
 /, 1-1, 4-10
 ;, 1-1
 [], 1-2
 ^, 4-11
 +, 4-10
 32 Bit Digital I/O Card, 3-6

A

Abort Motion, A-11
ABORTMASK Parameter, A-11
 ABS, 4-11
 Absolute Coordinates, 2-18, 2-39, 2-56, 2-57
 Absolute Dimension Programming Mode, 2-56, 2-58
 Absolute Position Programming, 2-56
 Absolute Position, Current, A-15
 Absolute Position, Determination of, A-15
 Absolute Programming Mode, 2-17, 2-39
 Absolute Value, 4-11
 Accel Mode Parameters, 2-49
ACCEL Parameter, A-6
 Accel Rate Parameter, 2-47, 2-49
 Accel Time Parameter, 2-43, 2-49
 Accel/Decel, 2-1, 2-8, 2-42
 Accel/Decel Control Group Box, 2-42, 2-43, 2-44, 2-47, 2-48
 Accel/Decel Rate Based, 2-49
 Accel/Decel Time Based, 2-49
 Accel/Decel Types, 2-42
ACCELERATE Parameter, A-8
 Acceleration, 2-12
Acceleration Feed Forward Gain Parameter, A-3
 Acceleration Loop, A-3
 Acceleration Mode, Linear, 2-46
 Acceleration Mode, Sinusoidal, 2-46
 Acceleration Rate, Set, 2-47
 Acceleration Rates, Setting, 2-50, 2-51
 Acceleration Time, Set, 2-43
 Acceleration, Instantaneous, 2-12
 Acceleration, Linear, 2-46, 2-49
 Acceleration, Sinusoidal, 2-43, 2-46, 2-49
 Acceleration/Deceleration, 2-1, 2-42
 AccelMode Parameter, 2-46
 ACCELMODE Parameter, A-7
 AccelRate Parameter, 2-42
 ACCELRATE Parameter, A-7
 AccelTime Parameter, 2-42
 Accuracy, A-3, A-9
 ACOS, 4-13
 Activate CDW Log File, 4-32
 Activate Cutter Compensation Right, 2-33
 Activate ICRC Left, 2-31
 Activate Normalcy Mode Left, 2-24
 Activate Normalcy Mode Right, 2-25
 Activate Right Cutter Compensation, 2-33
 Activated Feedhold, 2-19
 Active Level Parameter, 4-49
 Active, Fixture Offsets, 2-39
 Actual Position Data, 4-44
 Addition, 4-10
AFFGAIN Parameter, A-3
 Algorithm, Splined, 2-45
 Algorithm, splining, 2-17
 Allow Parameter Monitoring, 2-41
 Allow Safe Zone, 2-26, 2-27
 Analog Output Control Using PSOT Command, 5-2
 Analog/Digital Output Command (PSOT), 5-18
 And, 4-14
 AND, 4-14
 Application Program, A-2, A-3
 Arc Angle, 2-59, 2-60, 2-61
 Arc Generation, 2-8, 2-16, 2-21
 Arc Radius, 2-8, 2-59, 2-60, 2-61
 Arccosine, 4-13
 Arcsine, 4-13
 Arctangent, 4-13
 Array Element, 4-7
 Array Elements, 1-2
 Array Index, 4-7
 Array Indices, 1-2
 Array Size, 4-7
 Arrays, User, 4-7
 ASIN, 4-13
 Assign Symbolic Name, 4-5
 Asynchronous Commands, 4-1, 4-59
 ATAN, 4-13
 Autofocus command, 4-41
 Automated Motion, 2-18
 Autorun mode, 1-22
 Autorun.ini file, 1-22
 Aux Output Enable, A-11
AUX Parameter, A-3
 Auxiliary Output, A-11
 AUXMASK Parameter, A-3, A-10
AUXOFFSET Parameter, A-11
 Averaging Instantaneous Current, A-4
 AVGVEL Parameter (Read Only), A-2
AVGVELTIME Parameter, A-2
 Axes
 locking position counters, 5-11

Axes Plane Designation, 2-21
 Axes Planes, 2-21
 Axes Tool Path, 1-20
 Axes, Circular, 2-67, 2-68
 Axes, Linear, 2-67, 2-68
 Axes, Performance, 4-44
 Axes, Slave, 2-61
 Axis Acceleration, 2-42
 Axis Average Velocity, A-2
 Axis Deceleration, 2-42
 Axis Deceleration to 0 (Halt), A-10
 Axis Designations, 1-3, 4-41
 Axis Fault Screen, A-10
 Axis Fault, Changing Output State, A-3
 Axis Faults, Clearing, A-9
 Axis Faults, Viewing, A-9
 Axis Motion Disabled, 2-41
 Axis Motion, Terminated, 2-41
 Axis Naming, 4-41
 Axis Outputs, A-10
 Axis Outputs, Faults that Turn Off, A-10
 Axis Parameter Modification, 4-50
 Axis Parameter Reading, 4-48
 Axis Parameter Value, 4-48
 Axis Parameters, Valid, 2-41
 Axis Position, A-1
 Axis Processor, A-5, A-6, A-9
 Axis Processor Card, 2-45, 4-35, A-2, A-9
 Axis Processor I/O Lines, Active States, A-11
 Axis Processor, Testing Communications with, A-1
 Axis Safe Zone, Enabling and Disabling, A-6
 Axis Simulation Mode, A-6
 Axis Specification, 2-41
 Axis Status Screen, A-10
 Axis, Abort Mode, A-11
 Axis, B, 2-22
 Axis, Causing to Halt, A-10
 Axis, Clockwise Boundary of the Safe Zone, A-5
 Axis, Counter-clockwise Boundary of the Safe Zone, A-6
 Axis, Current Absolute Position of, A-15
 Axis, Enabling and Disabling Motor Torque of, A-3
 Axis, Establishing an Auxiliary Output for, A-3
 Axis, Faults that Disable, A-9
 Axis, Feed Rate Override, A-8
 Axis, Hard Names, 4-41
 Axis, Integral Gain of, A-2
 Axis, Keeping Stationary, A-3
 Axis, Linear, 2-61
 Axis, Master, A-12
 Axis, Motion Abort Faults, A-11
 Axis, Proportional Gain of, A-2
 Axis, Rotary, 2-22
 Axis, Spindle, 2-18, 2-61
 Axis, ThreadX, 2-18
 Axis, ThreadY, 2-18
 Axis, XPlane, 2-22

Axis, YPlane, 2-22

B

Backlash, Compensating for, A-3
 Backlash, Definition, A-3
 Backplane, VME, 3-13
 Ball Screw, Changing Direction, A-3
 Base Address, XYCOM, 3-13
base_speed Parameter, A-14
 B-Axis, 2-22
 B-Axis Initialization Dialog Box, 2-22
 BI, 3-14
 Binary Conversion, 4-14
 Binary I/O, 3-10
 Binary Input, 3-11
 Binary Input M-Code, 3-15
 Binary Inputs, 3-14
 Binary Output, 3-11
 Binary Output M-Code, 3-15
 Binary Outputs, 3-14
 Bit Mapping, 5-2, 5-12
 Bit Mask, A-9, A-10, A-11
 Bit Mask Control Window, A-10
 BIT#, 3-25
 Block Delete Character, 1-1
 Blocking Motion Commands, A-3
BLOCKMOTION Parameter, A-3
 BO, 3-14
 Board Number Parameter, 3-13
 Brackets, 4-7
 Brake, Motor, A-3

C

Call Library Subroutine, 4-26
 Call Subroutine, 4-26
 Called Program, 3-4
 CAM Points and Interpolation, A-12
 CAM Table, A-12, A-13
 CAM Table Execution Modes, A-13
 CAM Table Execution, using Absolute Slave Position Entries, A-13
 CAM Table Execution, using Relative Slave Position Entries, A-13
 CAM Table Position Types, A-12
 CAM Table, First Entry, A-12
 CAM Table, Master Position, A-12
 CAM Table, Offset to Master Position, A-12
 CAM Table, Slave Position, A-12
 CAM Tables, A-12
 Camming general, 4-56
 CAMOFFSET, 4-58
CAMOFFSET Parameter, A-12
 Cancel Fixture Offset, 2-39
 Cancel Parameter Monitoring, 2-41
 Card Number, XYCOM, 3-14

- CCW Boundaries, 2-26
- CCW Limit Switch, A-11
- CCW Motion, 2-9, 2-16
- CCWEOT Parameter, A-15
- CDW, 4-28
- CDW Log File, 4-32, 4-33
- CDW Log File, Deactivate, 4-33
- Changing Axis Parameters, 4-50
- Changing Output State on an Axis Fault, A-3
- Channel Number Parameter, 4-49
- Character strings, 1-9
- Circle Center Points, 1-8
- Circle Centerpoint, Relative Position, 1-8
- Circle Radius, 2-8
- Circles, 2-8, 2-16
- Circular Axes, 2-67, 2-68
- Circular Axis, 1-4
- Circular Interpolation, 2-31
- Circular Interpolation, 1-8, 2-8, 2-9, 2-16, 2-21, 2-33, 2-59, 2-60, 2-61
- Circular Interpolation, Inverse, 2-70
- Circular Interpolation, Normal, 2-69
- Circular Motion, 1-8
- Circular move, 2-5
- Circular moves, 2-23
- Circular Parameter, 2-67, 2-68
- CLLS, 4-26
- CLOCK Parameter**, A-2
- Clockwise Boundary of the Safe Zone, A-5
- Clockwise, Spindle, 3-3
- Close Custom Display Window, 4-29
- CLOSECDW, 4-29
- CLS, 4-26
- CNC Initialization Screen, 1-10, 1-12, 2-4, 2-35, 2-42, 2-43, 2-44, 2-47, 2-48, 2-50, 2-51, 2-56, 2-57, 2-64
- CNC Initialization Set-up Menu, 1-21
- CNC Initialization Set-up Screen, 1-7
- CNC Intercommunication
 - CNC parameters setup, 1-22
 - Global variables, 1-22
- CNC Manual Data Input Screen, 3-4
- CNC Number, 1-13
- CNC Parameters Plane Selection Menu, 1-8, 2-21
- CNC Preset command, 1-13
- CNC Processor, 4-24
- CNC Run Mode Screen, 3-4, 4-28, 4-29
- CNC Run Screen, 3-3
- CNC Time, 1-13
- Coefficients, Spline, 2-17
- Combine Parts Programs, 4-43
- Command Set, Extended, 4-1
- Command, Cycle Start, 3-4
- Command, GETPARM, 4-48
- Command, SETPARM, 4-50
- Command, STAT, 4-48
- Commanded Position Data, 4-44
- Commands, 5-2
- Comment Operator, 1-1
- Comments, 3-5
- Compensation for Backlash, A-3
- Completed Touch Probe Cycle, 4-49
- Condition Branch on Errors, 4-22
- Conditional Expression, 4-17
- Conditional Looping, 4-24, 4-25
- Conditional Statement, 4-19, 4-20
- Conditional Tracking, 5-2, 5-3
- CONFIGM command, 4-57
- Constant Acceleration vs. 1-Cosine, 2-42
- Constant Lead Thread Cutting, 2-18, 2-19
- Constant Surface Speed Spindle Programming, 2-63
- Contoured moves, 1-5, 2-5
- Contouring, Linear, 2-8
- Control button, Cycle Start, 2-19
- Control, Cycle Start, 3-3
- Control, Data Collection, 4-44, 4-47
- Control, Optional Stop, 3-3
- Controller, Programmable Logic, 1-14, 1-17
- Coordinated Motion, 2-42
- Coordinates, Absolute, 2-18, 2-39, 2-56, 2-57
- COS, 4-13
- Cosine, 4-13
- Counter Data
 - resetting, 5-3
 - retaining, 5-3
- Counter-clockwise Boundary of the Safe Zone, A-6
- Counterclockwise, Spindle, 3-3
- Creating a Numbering Scheme, 1-3
- Currently Active Fixture Offset, 2-39
- Custom Display Window, 4-28, 4-29, 4-32, 4-33
- Custom Display Window Commands, 4-28
- Custom Display Window, Close, 4-29
- Custom Display Window, Open, 4-28
- Cutter Compensation Axes, Set, 2-35
- Cutter Compensation Example**, 2-34
- Cutter compensation mode, 2-29
- Cutter Compensation Radius, Set, 2-35
- Cutter Compensation, Effect, 2-34
- Cutter Radius Compensation, Intersectional, 2-1, 2-28
- Cutter1 Pull-Down Menu, 2-35
- Cutter2 Pull-Down Menu, 2-35
- Cutting Cycle, Thread, 2-19
- Cutting Tool Number, 1-19
- Cutting Tool Orientation, 2-22
- Cutting Tool Radius, 2-28
- CW Boundaries, 2-26
- CW Limit Switch, A-11
- CW Motion, 2-8, 2-16
- CWEOT Parameter, A-15
- Cycle, 4-60
- Cycle Start Command, 3-4
- Cycle Start Control, 3-3
- Cycle Start, Button, 2-19
- Cycle, Thread Cutting, 2-19
- Cycle, Threading, 2-18

Cycles, Measuring Probe, 4-49

D

DATA, 4-44
 Data Acquisition, 4-44
 Data Acquisition Dialog Box, 4-45, 4-46
 Data Collection Control, 4-44, 4-47
 Data Collection Mode, 4-46
 Deactivate CDW Log File, 4-33
 Deactivate Cutter Compensation, 2-30
 Deactivating Tools, 1-19
 Debugging, A-6
DECEL Parameter, A-7, A-10, A-11
 Decel Rate Parameter, 2-48, 2-49
 Decel Time Parameter, 2-44, 2-49
DECELERATE Parameter, A-8
 Deceleration by Force, 2-14
 Deceleration Rate, Set, 2-48
DECELMODE Parameter, A-7
 DecelRate Parameter, 2-42
 DecelTime Parameter, 2-42
 DEFINE, 4-5
 Define a Safe Zone, 2-26
 Define Array, 4-7
 Define Entry Point, 4-8
 Define Subroutine, 4-8
 Define Symbolic Constant, 4-5
 Define User Variable, 4-6
 Defining User Arrays, 4-7
 Definition of Objects, 4-5
 Delay, 2-10
 DENT, 4-8
 Designating a Threading Axes, 2-18
 Designations, Axis, 4-41
 Detected Probe Input, 4-49
 DFS, 4-8
 Digital I/O Cards, XYCOM, 3-1, 3-13
 Digital Output Control Using PSOT Command, 5-2
 Digital Touch Probe Measuring, 4-49
 Digital/Analog Output Control Command (PSOT), 5-18
 Dimension Offset, 2-57
 Dimensions, Safe Zones, 2-27
 Direct RPM Spindle Programming Spindle Speed, 2-64
 Disable Feedrate Override, 3-4
 Disable Normalcy Mode, 2-24
 Disable Parameter Monitoring, 2-41
 Disable Safe Zones, 2-27
 Disable, Axis Motion, 2-41
 Disable, Spindle Feedrate Override, 3-4
 Disabled Mode of Operation, 2-24
DISABLEMASK Parameter, A-9
 Discontinue Cutter Compensation, 2-30
 DISPLAY, 4-29
 Distance Programming, 2-57

Distance Programming Mode, 2-56
 Distance, Incremental, 2-57
 Distance, Vectorial, 2-61
 Distances, Incremental, 2-18
 Distances, Setting, 2-50, 2-51
 Division, 4-10
 Dominant Feed Parameter, 2-67
 Dominant Feedrate Overview, 2-65
 DominantFeed Mode, 1-4
 Downloading Data, 5-2
 Drive Enable, A-11
 Drive Fault, A-11
DRIVE Parameter, A-3
 Dummy Parameters, A-1
 DVAR, 4-6, 4-7
 DVAR command, 1-10
 Dwell, 2-10

E

E, 1-7
 E word, 1-5, 1-7, 2-5
ECHO Parameter, A-1
 Effect of Cutter Compensation, 2-34
 Effect of Splining Algorithm, 2-17
 Element, Array, 4-7
 ELSE, 4-19
 Enable a Safe Zone, 2-26
 Enable Feedrate Override, 3-4
 Enable Parameter Monitoring, 2-41
 Enable Safe Zone, 2-26, 2-27
 Enable Safe Zones, 2-26
 Enable Spindle Feedrate Override, 3-4
 Enable/Disable Position Synchronized Output Firing (PSOF Command), 5-9
 End Extended Command Block, 1-2
 ENDIF, 4-19
 ENDM command, 4-34, 4-57
 ENDWHILE, 4-24
 English Units, 2-50, 2-51
 Entering Softnames, 1-3
 Entry Block, User Defined, 4-18
 Entry Field, Axis Name, 4-41
 Entry Field, Home Type, 4-35
 Entry Field, Tool Diameter, 2-35
 Entry Point, Define, 4-8
 Entry Point, Jump, 4-18
 EQ, 4-15
 Equal To, 4-15
 Error Tracking, Position, A-10
 Error, ICRC Activate Left, 2-31
 Error, Right Cutter Compensation Activate, 2-33
ERROR_TEXT, 3-25
 Errors, A-4, A-5, A-9
 Exclusive Or, 4-14
 EXECUTE, 4-27
 Execute OS/2 Program, 4-27

Execution, Stop, 3-3
 Exponentiation, 4-11
 Expression, 4-17
 Expression Evaluation, 4-12
 Extended Command Block, End, 1-2
 Extended Command Block, Start, 1-1
 Extended Command Set, 4-1

F

F, 1-4
 F word, 1-5, 1-7, 2-5
 Fast Feedrate, 2-7
 Fault Bit Mask, A-9
 Fault Condition, A-3
 Fault Handling, 2-26
FAULT Parameter, A-9
 Fault, Safe Zone, 2-26
 FAULT_LEVEL, 3-25
 FAULTMASK Parameter, A-3, A-9, A-10, A-11
 FAULTMSG.INI file, 3-25
 Faults, A-4, A-5, A-6, A-9, A-10, A-11
 Faults that Turn Off Axis Aux Outputs, A-10
 Faults Used to Disable Axis, A-9
FBWINDOW Parameter, A-5
 Feature, Enable Safe Zone, 2-26
 FEDM command, 4-58, 4-59
 Feed Per Min. Feedrate Programming, 2-60
 Feed Per Spindle Rev. Feedrate Programming, 2-61
 Feed Rate Override, A-8
 FEEDBACK Fault, A-5
 Feedback Resolution, A-9
 Feedhold Condition, 2-19
 Feedhold, Activated, 2-19
 Feedrate, 1-7, 2-59, 2-60
 Feedrate limiting, 1-5
 Feedrate Mode Programming, 2-59, 2-60, 2-61
 Feedrate Override Lock, 3-4
 Feedrate Override Unlock, 3-4
 Feedrate Override, Disable, 3-4
 Feedrate Override, Enable, 3-4
 Feedrate, Fast, 2-7
 Feedrate, Linear Axes, 2-19
 Feedrate, Linear Dominant, 2-68
 Feedrate, Rapid, 2-7
 Feedrate, S, 3-3
 Feedrate, Vectorial, 2-8, 2-10
FEEDRATEMODE Parameter, A-8
 Feedrates, Setting, 2-50, 2-51
 EOF command, 4-53
 Field Service Policy, 2-1
 File Close Command, 4-54
 File Open Command, 4-53
 File Operation Commands, 4-1, 4-53
 File Read Command, 4-53, 4-54
 File Reset Command, 4-54
 File Write Command, 4-55

File, MCODEX.INI, 3-15
 File, MODSCAN.INI, 3-9, 3-11, 3-13
 FILENAME, 3-26
 Filenames, 1-9
 Firing Distance
 calculations using multiple axes, 5-8
 maximum, 5-7
 Firing Distance Command, 5-6
 Firing Distance Entry, 5-2
 Fixture Offset, 2-39
Fixture Offset Example, 2-40
 Fixture Offset#1, 2-39
 Fixture Offset#2, Setting, 2-40
 Fixture Offset, Canceled, 2-39
 Fixture Offset, New, 2-40
 Fixture Offset, Old, 2-40
 Fixture Offsets, 2-40
 Floating Point, 4-15, 4-50
 Floating point constants, 1-8
 Flow, Parts Program, 3-1
 Following Error, Minimizing, A-2
 Force Deceleration, 2-14
 Format, Objects, 4-5
 FRAC, 4-11
 Fraction, 4-11
 FREE command, 4-34

G

G0, 1-5, 2-7
 G01, 2-8
 G02, 2-8, 2-16
 G03, 2-9, 2-16
 G04, 2-10
 G110, 2-69
 G111, 2-70
 G130, 2-71
 G131, 2-71
 G17, 2-21
 G18, 2-21
 G19, 2-21
 G20, 2-24
 G21, 2-24
 G22, 2-25
 G30, 2-17
 G33, 2-18
 G36, 2-26
 G37, 2-27
 G40, 2-30
 G41, 2-31
 G42, 2-33
 G43, 2-35
 G44, 2-35
 G45 command, 2-36
 G46 command, 2-36
 G47 command, 2-37
 G51, 4-49

G53, 2-39
 G54, 2-39
 G55, 2-40
 G56, 2-41
 G57, 2-41
 G60, 2-43
 G61, 2-44
 G62, 2-45
 G63, 2-46
 G64, 2-46
 G65, 2-47
 G66, 2-48
 G67, 2-49
 G68, 2-49
 G70, 2-50
 G71, 2-51
 G8, 2-12
 G9, 2-14
 G90, 2-56
 G91, 2-57
 G92, 2-58
 G93, 2-59
 G94, 2-60
 G96, 2-63
 G97, 2-64
 G98, 2-67
 G99, 2-68
 G-Code Description, 2-1
 G-Codes Pull-Down Menu, 2-4, 2-42, 2-50, 2-51, 2-56, 2-57
 GE, 4-16
 General Purpose Timer, A-2
 Generation, Arc, 2-8, 2-16, 2-21
 Generation, Circular, 2-8, 2-16
 Generator, Trajectory, 2-42
 GETPARM, 4-48
 GETPARM Command, 4-48
 GLBALIAS.INI, 1-11
 Global alias file, 1-11
 Global Data, 3-11
 Global variables, 1-10
 Go To User Defined Entry Block, 4-18
 Greater Than, 4-16
 Greater Than or Equal To, 4-16
 GT, 4-16

H

Halt Program, 3-3
 Halt Spindle Movement, 3-3
HALTMASK Parameter, A-10
 HANDwheel command, 4-34
 Hard Names, 4-41
 Hardnames, 1-3
 HARDNAMES, 4-41
 Hardware Home, 4-35
 Hardware Home Position, 2-58, A-5, A-6

Helical Interpolation, 2-9
 Hexidecimal integers, 1-9
 Holding Areas, Temporary, A-1
 HOME, 4-35
 Home (Hardware) Position, A-5, A-6
 Home Feedrate, A-9
 Home Limit and Marker Pulse, Minimum Distance Between, A-9
 Home Limit Switch, A-11
 Home Limits Switch, A-9
 Home Marker Pulse, A-9
 Home Position, A-9, A-15
 Home, Hardware, 4-35
 Home, Position, 2-58
 HOME_SWITCH_TOLERANCE Fault, A-9
HOMESWITCHTOL Parameter, A-8
 Homing Process, Completion of, A-15
 Homing Sequence Accuracy, A-9
 Host, 1-14, 1-17

I

I/J/K, 1-8
 I/O, 3-5
 I/O Bit, Virtual, 3-10
 I/O Points, Virtual, 3-15
 I/O_TYPE, 3-25
IAVGLIMIT Parameter, A-4
IAVGTIME Parameter, A-4
 ICRC, 1-19, 2-1, 2-28
 ICRC, Activate Left, 2-31
 ICRC, Cutter Compensation Activate Right, 2-33
 ICRC, Deactivate, 2-30
 ICRC, Tool Path, 2-34
 If-Then-Else-EndIf Statement, 4-19
 Ignoring Motion Commands, A-3
IMAX Parameter, A-3, A-4
 Implement a Safe Zone, 2-26
 In Position Band, A-5
 In Position Status Bit, A-5
 Inch Dimension Programming Mode, 2-50
 INCLUDE, 4-43
 Included Program File, 4-43
 Incremental Distance, 2-57
 Incremental Distance Mode, 2-39, 2-40
 Incremental Distances, 2-18
 Incremental Position Programming, 2-57
 Incremental Programming Mode, 2-17
 INDEX statement, 4-59
 Index, Array, 4-7
 Indexing Array Elements, 1-2
 Indicate Array Size, 4-7
 Initialization of the System, A-1
 Initialize Local Variable, 1-12
 Initializing Binary Input M-Codes, 3-18
 Initializing Binary Output M-Codes, 3-20, 3-21
 Initializing Binary Outputs, 3-20

Initializing Register Input M-Codes, 3-22
 Initializing Register Output M-Codes, 3-23
 Initializing Register Outputs, 3-24
 Initializing Variables, 4-6
 Initializing Virtual I/O, 3-25
 Initiate a Home, 4-35
 Initiating Data Acquisition, 4-45
INPOSLIMIT Parameter, A-4
 Input, 5-4
 Input, Binary, 3-11
 Input, Probe, 4-49
 Input/Output, 3-5
 Inputs, 3-1, 3-5, 3-10, 3-13, 3-14, A-11
 "i" argument, 5-4
 conditional tracking based on the states of, 5-3
 Inputs, Binary, 3-14
 Instantaneous Acceleration, 2-12
 Instantaneous Current, Averaging, A-4
 Instantaneous Current, Averaging, A-4
 Instantaneous Speed, Maximum, A-5
 INT, 4-11
Integral Gain (KI) Parameter, A-2
 Integral Gain of the Velocity Loop, A-2
 Interpolation and CAM Points, A-12
 Interpolation, Circular, 2-8, 2-9, 2-16, 2-21, 2-31, 2-33, 2-59, 2-60, 2-61
 Interpolation, Circular Inverse, 2-70
 Interpolation, Helical, 2-9
 Interpolation, Linear, 2-8, 2-9
 Interpolation, Normal Circular, 2-69
 Interrupt, Program, 3-3
 Interrupt, System, A-10
 Intersection Cutter Radius Compensation, 2-1, 2-28
 Intersectional Cutter Radius Comp., 1-19
INTMASK Parameter, A-9
 Inverse Circular Interpolation, 2-70
 Inverse Time Feedrate Programming, 2-59
IOLEVEL Parameter, A-10

J

Joining Parts Programs, 4-43
 JUMP, 4-18
 Jump Entry Point, 4-18
 Jump to User Defined Enter Block, 4-18
 Jump to User Defined Entry Block, 4-18

K

Keyword "CLOSE", 4-46
 Keyword "Do", 4-24
 Keyword "E", 2-67
 Keyword "ENDWHILE", 4-24
 Keyword "F", 2-8, 2-10, 2-17, 2-61, 2-68
 Keyword "GLOBAL", 3-11
 Keyword "OPEN", 4-44, 4-45
 Keyword "P", 2-26

Keyword "PLC", 3-6, 3-9, 3-10
 Keyword "R", 2-63
 Keyword "S", 2-64, 3-3
 Keyword "SF", 2-63
 Keyword "START", 4-45, 4-46
 Keyword "STOP", 4-46
 Keyword "T", 1-19
 Keyword "THEN", 4-19
 Keyword "XYCOM", 3-6, 3-13, 3-14
 Keyword, F, 2-8
 Keyword, Feedrate, 1-4, 1-7
 Keyword, Spindle Speed, 1-7
 Keyword, Tool Word, 1-19
 Keywords "I, J, K", 1-8
 Keywords "I/J/K", 2-69, 2-70
 Keywords, "I/J/K", 2-21
KI Parameter, A-2
KP Parameter, A-2

L

Label, Define, 4-8
 Labels, 4-5
 Laser Pulse Output Definition, 5-2
 LE, 4-16
 Lead off move, 2-32
 Lead on move, 2-32
 Left Parenthesis, 4-8
 Left, Path Compensation, 2-31
 Less Than, 4-16
 Less Than or Equal To, 4-16
 Library Subroutine, 4-37
 Line Numbers, 1-3
 Linear Acceleration, 2-46, 2-49
 Linear Acceleration Mode, 2-46
 Linear Axes, 1-7, 2-67, 2-68
 Linear Axes Feedrate, 2-19
 Linear Axes Move, 2-19
 Linear Axis, 1-4, 2-61
 Linear Contouring, 2-8
 Linear Feedrate Dominant, 2-68
 Linear Interpolation, 2-8, 2-9
 Linear move, 2-5
 Linear Move, 2-21
 Linear parameter, 1-7
 Linear Parameter, 2-67, 2-68
 Linear Ramping, A-7
 Linear Threads, 2-18
 Linear Trajectory, 2-42
 Linear-vs-Sinusoidal Option, 2-42
 Local variables, 1-10
 Locate Axis(s), 4-35
 Locate Part in Space, 4-49
 Lock Spindle Feedrate Override, 3-4
 Lock, Feedrate Override, 3-4
 Log File, 4-32, 4-33
 Logical Operators, 4-14

Loop, Conditional, 4-24, 4-25
 Loop, Repeat, 4-21
 Loop, While, 3-4, 4-25
 LT, 4-16

M

M0, 3-3
 M01, 3-3
 M03, 3-3
 M04, 3-3
 M05, 3-3
 M19, 3-3
 M30, 3-4
 M47, 3-4
 M48, 3-4
 M49, 3-4
 M50, 3-4
 M51, 3-4
 Machine Parameter Pull-Down Menu, 4-35, 4-41, 4-42
 Machine Parameter Set-up Screen, 1-4, 2-67, 2-68
 Machine Parameters Set-up Screen, 1-7
 Machine Position Registers, 2-39, 2-40, 2-58
 Main Menu, A-2, A-10
 Main Program File, 4-43
 Manual Feedrate Override, 2-7
 Master Axis, A-12
 Master Position, A-11
 Master Position, Nonzero, A-12
 Master Positions, Offsets, A-12
 MASTERLENGTH, 4-58
MASTERPOS Parameter, A-11, A-12
 Master-slave CNC commands, 4-1, 4-56
 max_phase Parameter, A-14
 M-Code, 3-1
 M-Code Programming, 3-5
 M-Code Variables, 3-5
 MCODEx.INI File, 3-15
 Measuring Cycles, Probe, 4-49
 Memory Mapped I/O, 3-5
 Memory, I/O, 3-5
 Menu Option, Linear-vs-Sinusoidal, 2-42
 Menu Option, Time-vs-Rate Based, 2-42
 Merge Parts Programs, 4-43
 Method, Splining, 2-17
 Metric Dimension Programming Mode, 2-51
 Metric Units, 2-50, 2-51
 MFO, 1-6
 M-Functions vs. Virtual I/O, 3-1, 3-15
 Minimizing Machine Time, 2-7
 Miscellaneous Commands, 4-41
 Miscellaneous Functions, 4-1
 MOD, 4-10
 Modal G-Codes, 2-1
 Modbus Plus Network, 3-9
 Mode, 5-6, 5-9, 5-14, 5-17, 5-18

Mode 0, 4-57
 Mode 1, 4-57
 Mode 2, 4-57
 Mode of Operation Disabled, 2-24
 Mode, Activate Normalcy Right, 2-25
 Mode, Incremental Distance, 2-39, 2-40
 Mode, Linear Acceleration, 2-46
 Mode, Normalcy, 2-1, 2-22
 Mode, Normalcy Disabled, 2-24
 Mode, Normalcy Left, 2-24
 Mode, Splining, 2-17
 Modicon Programmable Logic Controllers, 3-6
 Modify Variables, 4-6
 Modifying Axis Parameters, 4-50
 Modscan Thread, 3-5, 3-10, 3-14
 MODSCAN.INI file, 3-9
 MODSCAN.INI File, 3-11, 3-13
 Modulus, 4-10
 Monitor axis speed command, 4-50
 MONSPD command, 4-50
 Motion Cycle, Automated, 2-18
 Motion Status, 1-14
 Motion, CCW, 2-9, 2-16
 Motion, Circular, 1-8
 Motion, Coordinated, 2-42
 Motion, CW, 2-8, 2-16
 Motion, smooth, 2-17
 Motion, Spline, 2-17
 Motor Brake, Activation of, A-3
Motor Torque, Enabling and Disabling, A-3
 Motor Torque, Steady, A-6
 Motor Velocity, A-14
 Move, Linear, 2-21
 Move, Spline, 2-17
 Movement, point-to-point, 2-7
 MOVETO statement, 4-60
 Multiple CNCs, 1-22
 Multiplication, 4-10

N

Name, Symbolic, 4-5
 Naming, Axes, 4-41
 NE, 4-15
 Negation, 4-14
 Nested Called Program, 3-4
 Nested Repeat Loops, 4-21
 Nested While Loop, 4-25
 New Fixture Offset, 2-39, 2-40
 Non-Modal G-Codes, 2-1
 Normal Circular Interpolation, 2-69
 Normalcy alignment, 2-23
 Normalcy Mode, 2-1, 2-22
 Normalcy Mode Left, Activate, 2-24
 Normalcy Mode Right, Activate, 2-25
 Normalcy speed limit, 1-5
 NOT, 4-14

Not Equal To, 4-15
Nxxxx, 1-3

O

Objects, Definition Of, 4-5
Offset Values of Tools, 1-20
Offset#1, Fixture, 2-39
Offset, Dimension, 2-57
Offsets to Master Axis Position, A-12
Old Fixture Offset, 2-39, 2-40
Omitting a Program Block, 1-1
ONERRGOTO, 4-22
Open Custom Display Window, 4-28
OPENCDW, 4-28
Operator, Absolute Value, 4-11
Operator, Addition, 4-10
Operator, And, 4-14
Operator, Arccosine, 4-13
Operator, Arcsine, 4-13
Operator, Arctangent, 4-13
Operator, Cosine, 4-13
Operator, Division, 4-10
Operator, Equal To, 4-15
Operator, Exclusive Or, 4-14
Operator, Exponent, 4-11
Operator, Fraction, 4-11
Operator, Greater Than, 4-16
Operator, Greater Than or Equal To, 4-16
Operator, Less Than, 4-16
Operator, Less Than or Equal To, 4-16
Operator, Modulus, 4-10
Operator, Multiplication, 4-10
Operator, Negation, 4-14
Operator, Not Equal To, 4-15
Operator, Or, 4-14
Operator, Precedence, 4-12
Operator, Rounding, 4-11
Operator, Shift Left, 4-15
Operator, Shift Right, 4-15
Operator, Sine, 4-13
Operator, Square Root, 4-11
Operator, Subtraction, 4-10
Operator, Tangent, 4-13
Optional Stop, 3-3
Optional Stop Control, 3-3
Or, 4-14
OR, 4-14
Orientation, Cutting Tool, 2-22
OS/2 Program Execution, 4-27
OSCillate command, 4-60
Output, 5-5
Output Definition of Laser Pulse, 5-2
Output Firing, Enabling and Disabling, 5-9
Output State, Changing on an Axis Fault, A-3
Output Toggle Mode, 5-15
Output, Auxiliary, A-11

Output, Binary, 3-11
Outputs, 3-1, 3-5, 3-10, 3-13, 3-14, A-10, A-11
"out_map" argument, 5-4
configuration of DACs to provide position ramping, 5-19
control using PSOT command, 5-2
controlling states based on inputs, 5-3
PSOT Command, 5-18
setting output voltages of DACs using PSOT, 5-18
setting states of, 5-3
setting using PSOT command, 5-18
Outputs OUT0-OUT15
setting states of, 5-3
Outputs, Binary, 3-14
Over Current Operation, Detection of, A-4
Override, Feedrate Lock, 3-4
Override, Spindle Speed, 2-19
Overriding Feed Rate, A-8
Overview, Accel/Decel, 2-1, 2-42
Overview, Dominant Feedrate, 2-65
Overview, ICRC, 2-1, 2-28
Overview, Normalcy Mode, 2-1, 2-22

P

Parameter Monitoring, 2-41
Parameter Monitoring, Disable, 2-41
Parameter Monitoring, Enable, 2-41
Parameter Monitoring, Simultaneously, 2-41
Parameter types, 1-8
Parameter, Accel Rate, 2-47, 2-49
Parameter, Accel Time, 2-43, 2-49
Parameter, AccelMode, 2-46
Parameter, AccelTime, 2-42
Parameter, Active Level, 4-49
Parameter, Board Number, 3-13
Parameter, Channel Number, 4-49
Parameter, Decel Rate, 2-48, 2-49
Parameter, Decel Time, 2-44, 2-49
Parameter, DecelTime, 2-42
Parameter, Dominant Feed, 2-67
Parameter, Ramp Type, 2-46, 2-47, 2-48, 2-49
Parameter, Rate Based, 2-47, 2-48
Parameter, SafeZone CCW, 2-26
Parameter, SafeZone CW, 2-26
Parameter, SafeZoneMode, 2-26
Parameter, Size, 3-14
Parameter, Spindle Speed, 3-3
Parameter, Thread Taper, 2-18
Parameter, Time Based, 2-43, 2-44, 2-46
Parameter, Time-vs-rate Based, 2-42
Parameter, Tool Radius, 2-35
Parameter, Virtual I/O Point, 3-14
Parameter, VME Address, 3-13
Parameter, XPlane, 2-22
Parameter, YPlane, 2-22
Parameters, Accel Mode, 2-49

- Parameters, Plane Select, 2-69, 2-70
- Parameters, Ramp Type, 2-46
- Parameters, Rate Based, 2-47, 2-48, 2-49
- Parameters, Time Based, 2-46
- Parenthesis, 4-4
- Part Rotation Angle, 2-19
- Parts Program Flow, 3-1
- Parts Program, Acceleration, A-8
- Parts Program, Join, 4-43
- Parts Program, Specifying Deceleration Mode from, A-8
- Parts Program, Specifying Safe Zone Parameters from, A-5, A-6
- Parts Programs, Debugging, A-6
- Path Compensation Left, 2-31
- Path Compensation Right, 2-33
- Percent Sign, 1-1
- Permit Parameter Monitoring, 2-41
- Permit Safe Zone, 2-26, 2-27
- PGAIN Parameter**, A-2
- phase_speed Parameter, A-14, A-15
- Place Items In Window, 4-29
- Placing Data In User Stack, 4-37
- Plane Select Parameters, 2-69, 2-70
- Plane Selection, 2-1, 2-21
- Planes, Axes, 2-21
- PlaneSelect group, 2-21
- PLC, 3-1, 3-9
- PLC I/O vs. Virtual I/O, 3-10
- PLC Register Numbers, 3-10
- PLC Registers, 3-10
- Point, Entry, 4-8
- Points In Space, 2-39
- point-to-point move, splined, 2-17
- Point-to-point movement, 2-7
- POP, 4-38
- POS Parameter**, A-1
- POSERRLIMIT Parameter**, A-4
- Position Counter
 - clearing, 5-17
 - stopping, 5-17
- Position Counters
 - setting and locking, 5-10
- Position Error Limit Fault, A-4
- Position Error Tracking, A-10
- Position Error, Maximum, A-4
- Position Following Error, Minimizing, A-2
- Position Gain**, A-2
- Position Programming, Incremental, 2-57
- Position Ramping, 5-19
- Position Synchronized Output Firing Distance
 - Command, 5-6
- Position Synchronized Output Pulse Configuration (PSOP), 5-14
- Position Synchronized Output Using Bit Mapping (PSOM), 5-12
- Position Synchronized Output with Real-time Control, 5-17
- Position Tracking
 - based on inputs, 5-3
 - unconditional, 5-3
- Position, Hardware Home, 2-58, 4-35
- Positioning Accuracy, A-3
- Positioning, Point-to-point, 2-7
- Precedence, 4-12
- Predefined Axis Names, 4-41
- Preset Position, 2-39
- Preset Position Registers, 2-39, 2-40, 2-58
- Prevent Parameter Monitoring, 2-41
- Probe Input, 4-49
- Probe Measuring Cycles, 4-49
- Probe Polarity, 4-49
- Probe Touch, 4-49
- Processor Card, 4-35
- Program Block, 4-6
- Program Block, Activating, 1-1
- Program Block, Deactivating, 1-1
- Program Execution, Restart, 3-4
- Program Flow, 4-1, 4-6, 4-8
- Program Interrupt, 3-3
- Program Position Registers, 2-39
- Program Stop, 3-3
- Programmable Logic Controller, 1-14, 1-17
- Programmable Logic Controllers, 3-1, 3-9
- Programmed Position, 2-39
- Programming
 - commands, 5-2
- PROGRAMMING**, 5-1
- Programming Conventions**, 5-1
- Programming Mode, Absolute, 2-17, 2-39, 2-58
- Programming Mode, Absolute Dimension, 2-56
- Programming Mode, Distance, 2-56
- Programming Mode, Incremental, 2-17
- Programming Mode, Metric, 2-51
- Programming operators, 4-10
- Programming the Tool Path, 2-28
- Programming Units, 2-50
- Programming, Distance, 2-57
- Programming, Feed Per Min., 2-60
- Programming, Feed Per Spindle Rev. Feedrate, 2-61
- Programming, Feedrate Mode, 2-59
- Programming, M-Codes, 3-5
- Programming, Spindle Speed, 2-63, 2-64
- Proportional Gain (KP) Parameter**, A-2
- Proportional Gain of the Velocity Loop, A-2
- PSO Commands
 - case sensitivity, 5-1
- PSO Programming commands, 5-2
- PSOC, 5-4
- PSOC Command, 5-3
- PSOD, 5-7
- PSOD Command, 5-6
- PSOF, 5-10

PSOF Command, 5-9
 PSOM Command, 5-12
 PSOP, 5-15
 PSOP Command, 5-14
 PSOR Command, 5-17
 PSOT, 5-20
 PSOT Command, 5-18
 Pulse Configuration Command (PSOP), 5-14
 Pulse Gap Interval, 5-14
 Pulse Lead Time, 5-14
 Pulse Output

- absolute distances, 5-6
- fixed distances, 5-6
- incremental distances, 5-6

 Pulse Outputs, 5-6
 Pulse Ramp Up/Down Time, 5-14
 Pulse Trail Time, 5-14
 Pulse Train, 5-9

- components, 5-14
- defining, 5-14

 Pulse Width Time, 5-14
 PUSH, 4-37
Push/Pop Example, 4-38
 Put Data On User Stack, 4-37

R

Radio Freq. Tool I.D. No., 1-19
 Radius, Arc, 2-8
 Radius, Circle, 2-8
 Radius, Cutting Tool, 2-28
 Ramp Type Parameter, 2-46, 2-47, 2-48, 2-49
 Ramp Type Parameters, 2-46
 Ramping, A-8
 Ramping, Linear, A-7
 Ramping, Rate-based, A-7
 Ramping, Selection of, A-7
 Ramping, Sinusoidal, A-7
 Ramping, Time-based, A-7
 Range, Array Index, 4-7
 Range, Virtual I/O Points, 3-14
 Rapid Feedrate, 2-7
 Rapid Motion, 2-12
 Rate Based Parameter, 2-47, 2-48
 Rate Based Parameters, 2-47, 2-48, 2-49
 Rate-based Linear Ramping, A-7, A-8
 Rate-Based Ramping, A-7
 Rate-based Sinusoidal Ramping, A-7, A-8
 Read Cycle, Touch Probe, 4-49
 Readable Axis Parameters, 2-41
 Reading Axis Parameters, 4-48
 Real Time Tracking Control, 5-2
 Real-time Control (PSOR), 5-17
 RECORDOFF, 4-33
 RECORDON, 4-32
 REF, 4-35
 Register I/O, 3-10

Register Input M-Code, 3-15
 Register Output M-Code, 3-15
 Registers, Machine Position, 2-39, 2-40, 2-58
 Registers, PLC, 3-10
 Registers, Preset Position, 2-39, 2-58
 Registers, Preset Position, 2-40
 Registers, Program Position, 2-39
 Relational Operators, 4-15
 Remote Host, 1-14, 1-17
 Remove Data From User Stack, 4-38
 Repeat Loop, 4-21
 Repeat Loops, Nested, 4-21
 Restart Program Execution, 3-4
 Restrict Safe Zones, 2-27
 Restricted Areas, 2-27
 Retract Position, 2-19
REVERSALMODE Parameter, A-3
 Right Cutter Compensation, Activate, 2-33
 Right Parenthesis, 4-8
 Right, Path Compensation, 2-33
 RMS Current Limit Fault, A-4
 Rotary Axis, 2-22
 Rotary Feedrate Dominant, 2-67
 Rotary parameter, 1-7
 Rotation Angle of Threads, 2-19
 Rounding, 4-11
 RPT, 4-21

S

S, 1-7
 Safe Zone, A-15
 Safe Zone Boundaries, 2-27
 Safe Zone Fault, 2-26, 2-27, A-6
 Safe Zone Restrictions, 2-26
 Safe Zone, Clockwise Boundary, A-5
 Safe Zone, Counter-clockwise Boundary, A-6
 Safe Zone, Disable, 2-27
 Safe Zone, Enable, 2-26, 2-27
 Safe Zone, Enabling and Disabling, A-6
 Safe Zones, A-15
 Safe Zones, Enable, 2-26
 SafeZone CCW Parameter, 2-26
 SafeZone CW Parameter, 2-26
SAFEZONECCW Parameter, A-5, A-6
SAFEZONECW Parameter, A-5, A-6, A-15
 SafeZoneMode Parameter, 2-26
 SAFEZONEMODE Parameter, A-5, A-6, A-15
 Select Plane, 2-21
 Select, Plane, 2-1, 2-21
 Selecting, Threading Axes, 2-18
 Semicolon, 1-1, 3-15
 Semicolon, Special Symbol, 3-5
 Servo Spindle Axes, 3-3
 Set Acceleration Rate, 2-47
 Set Acceleration Time, 2-43
 Set Cutter Compensation Axes, 2-35

- Set Cutter Compensation Radius, 2-35
- Set Deceleration Rate, 2-48
- Set Fixture Offset #2, 2-40
- Set Profile Time, 2-45
- SETPARM, 4-50
- SETPARM Command, 4-50
- Setting Acceleration Rates, 2-50, 2-51
- Setting Fixture Offsets, 2-39
- Shift Left, 4-15
- Shift Right, 4-15
- SHL, 4-15
- SHR, 4-15
- Simulated Axis, A-6
- Simulation Mode, A-6
- SIMULATION Parameter**, A-6
- Simultaneous Parameter Monitoring, 2-41
- SIN, 4-13
- Sine, 4-13
- Single Pulse, Defining, 5-14
- Sinusoidal Acceleration, 2-43, 2-46, 2-49
- Sinusoidal Acceleration Mode, 2-46
- Sinusoidal Ramping, A-7
- Size Parameter, 3-14
- Size, Array, 4-7
- Slave Axes, 2-61
- Slave Position CAM Table Entries, Absolute, A-13
- Slave Position CAM Table Entries, Relative, A-13
- Slave Profile Execution, A-12
- SLEW command, 4-36
- Slope of a Velocity Curve, **A-14**
- Smooth Motion, 2-17
- Smooth Point-to-point Move, 2-17
- SOFTLIMITMODE Parameter**, A-15
- Softnames, 1-3
- SOFTNAMES, 4-41
- Software Home, 2-58
- Software Limit, A-15
- Software Limits, A-15
- Software Limits Activation Mode, A-15
- Space, Locate Part, 4-49
- Special Symbol, \$, 3-5
- Special Symbol, (, 4-8
- Special Symbol, (), 4-4
- Special Symbol,), 4-8
- Special Symbol, :, 3-5, 3-15
- Special Symbol, [], 4-7
- Special Symbol, Semicolon, 3-15
- Specify Array Size, 4-7
- Specify Cutter Compensation Axes, 2-35
- Specify Cutter Compensation Radius, 2-35
- Specify Fixture Offset, 2-39
- Specify Label, 4-8
- Specify Subroutine, 4-8
- Speed, Spindle, 2-19, 2-61, 2-63, 3-3
- Speeds, Setting, 2-50, 2-51
- Spindle Axis, 1-7, 2-18, 2-61
- Spindle Feedrate Override, Disable, 3-4
- Spindle Feedrate Override, Enable, 3-4
- Spindle Feedrate Override, Lock, 3-4
- Spindle Feedrate Override, Unlock, 3-4
- Spindle Movement, Stop, 3-3
- Spindle Off, 3-3
- Spindle Off/Re-orient, 3-3
- Spindle On Clockwise, 3-3
- Spindle On Counterclockwise, 3-3
- Spindle Pull-Down Menu, 2-64
- Spindle rotary axes speed, 1-7
- Spindle Speed, 2-19, 2-61, 2-63, 2-64, 3-3
- Spindle Speed "S", 1-7
- Spindle Speed Override, 2-19
- Spindle Speed Parameter, 3-3
- Spindle Speed Programming, 2-63, 2-64
- Spindle Velocity, 2-19
- Spline Move, 2-17
- Splined Motion, 2-17
- Splined Moves, 2-17
- Splining Algorithm, 2-17, 2-45
- Splining Method, 2-17
- Splining Mode of Operation, 2-17
- SQRT, 4-11
- Square Root, 4-11
- Stack, 4-37
- Stack Pointer, 4-37, 4-38
- Start Extended Command Block, 1-1
- Starting Angle, Thread, 2-19
- STAT Command, 4-48
- Static variables, 1-12
- STATUS Parameter, A-11
- Stop Cutter Compensation, 2-30
- Stop Parameter Monitoring, 2-41
- Stop Spindle Movement, 3-3
- Stop, Optional, 3-3
- Stop, Program, 3-3
- STRM command, 4-61
- Subroutine Call, 4-8
- Subroutine Name, 4-5
- Subroutine, Call, 4-26
- Subroutine, Define, 4-8
- Subroutine, Library, 4-37
- Subroutine, Library Call, 4-26
- Subroutines, 4-5
- Subtraction, 4-10
- Summary, Extended Commands, 4-1
- Symbol, (, 1-1
- Symbol,), 1-2
- Symbolic Constant, 4-5
- Symbolic Name, 4-5
- Symbols, /, 1-1
- Symbols, [and], 1-2
- Symbols, Comment, 1-1
- Symbols, N, 1-3
- Symbols, Percent Sign, 1-1
- Symbols, Semicolon, 1-1
- SYNC command, 4-57

Sync Table Control, A-3
 Synchronization, 5-6
 Synchronization, Disabled mode of, A-12
 Synchronized Auxiliary Output Tables, A-11
 Synchronized Contouring, 2-8
 Synchronized Motion, A-12, A-13
 Synchronized Motion, CAM Tables, A-12
SYNCSPEED Parameter, A-12
 System I/O, 3-5
 System Initialization, A-1
 System Interrupt, Faults that Cause a, A-10
 System Variables, 1-13

T

Take Data From User Stack, 4-38
 TAN, 4-13
 Tangent, 4-13
 Taper, 2-19
 Tapered Threads, 2-18
 Temporary Holding Areas, A-1
 Terminate Axis Motion, 2-41
 Terminate Parameter Monitoring, 2-41
 Testing Communications with the Axis Processor, A-1
 Thread Cutting Cycle, 2-19
 Thread Cutting, Constant Lead, 2-18, 2-19
 Thread Lead, 2-18, 2-19
 Thread Starting Angle, 2-19
 Thread Taper Angle, 2-18, 2-19
 Thread, Modscan, 3-5, 3-10, 3-14
 Thread, Rotation Angle, 2-19
 Thread, Starting Angle, 2-19
 Threading Axes Designations, 2-18
 Threading Cycle, 2-18
 Threads, in definition, 2-18
 Threads, Linear, 2-18
 Threads, Tapered, 2-18
 ThreadX Axis, 2-18
 ThreadY Axis, 2-18, 2-64
 Time Based Parameter, 2-43, 2-44, 2-46
 Time Based Parameters, 2-46
 Time Based, Parameter, 2-46
 Time, Dwell, 2-10
 Time-based Linear Ramping, A-7, A-8
 Time-based Ramping, A-7
 Time-based Sinusoidal Ramping, A-8
 Timer, General Purpose, A-2
 Time-vs-Rate Based Option, 2-42
 Time-vs-rate Based Parameter, 2-42
 Toggle Mode for Outputs, 5-15
 Tool Deactivation, 1-19
 Tool Diameter Entry Field, 2-35
 Tool Differentiating, 1-19
 Tool File, 1-19
 Tool Orientation, 2-22
 Tool Path Programming, 2-28
 Tool Radius Parameter, 2-35

Tool Records, 1-19
 Tool Word, 1-19
 Tool, Actual Radius, 1-20
 Tool, Boring Bar Number, 1-20
 Tool, Holder Number, 1-20
 Tool, Inspection Station, 1-21
 Tool, Inspection Station Probe Set Data, 1-21
 Tool, Location, 1-19
 Tool, Nominal Radius, 1-20
 Tool, Number Passes, 1-20
 Tool, Orientation, 1-20
 Tool, System Entry Date, 1-20
 Tool, System Entry Time, 1-21
 Tool, Tip, 2-22
 Tool, Type, 1-19
 Tool, X-Axis Offset, 1-20
 Tool, Z-Axis Offset, 1-20
 Tools with Different Diameters, 2-28
 Torque Command, A-6
 Torque Command Data, 4-44
 Torque used as Feedback, A-6
Torque, Enabling and Disabling, A-3
 Torque, Steady, A-6
 Touch Probe Measuring, Digital, 4-49
 Touch Probe Read Cycle, 4-49
 Tracking Axes, Specifying, 5-2
 Tracking Control, Real Time, 5-2
 Trajectory Generator, 2-42
 Trajectory, Linear, 2-42
 Traps, A-5
 Trigonometric Operator, 4-13
 Txxxx, 1-19

U

UNIDEX 31, Axis Processor Card, A-2
 Units, English, 2-50, 2-51
 Units, Metric, 2-50, 2-51
 Units, Programming Mode, 2-50, 2-51
 Unlock Spindle Feedrate Override, 3-4
 Unlock, Feedrate Override, 3-4
 User Defined Entry Block, Jump, 4-18
 User Defined M-Functions, 3-5
 User Stack, 4-26, 4-37, 4-38
 User Stack Data Placement, 4-37
 User Stack, Data Removal, 4-38
 User Variable, 4-6

V

Valid Axis Parameters, 2-41
 Value, Axis Parameter, 4-48, 4-50
 Variable Allocation Group Box, 1-10, 1-12
 Variable Name, 4-5
 Variable Usage, 4-6
 Variable V0-V255, 5-12
 Variable, User, 4-6

Variables, 1-9, 3-5, 4-5
 Variables V0-V255, 5-6
 size in bytes, 5-12
 Vectorial Distance, 2-61
 Vectorial Feedrate, 2-8, 2-10, 2-17
 Velocity, 2-12, 2-14
 Velocity Command Integration, A-5
 Velocity Command/Feedback Integration, Max
 Difference, A-5
 Velocity Curve Slope, **A-14**
 Velocity Error Free System, A-6
Velocity Feed Forward Function, A-2
 Velocity Feedback Integration, A-5
 Velocity Profile with G9, 2-15
 Velocity Profile without G9, 2-14
 Velocity Profiling, 2-45
 Velocity Ramping, 5-18
 Velocity Ramping, Selection of, A-7
 Velocity Trap, A-5
 Velocity, Changing, A-7
 Velocity, Spindle, 2-19
 VELTIMECONST, 2-13
VELTRAP Parameter, A-5
VFF Parameter, A-2
 Virtual I/O, 3-5
 Inputs, 3-7
 Outputs, 3-7
 Virtual I/O Bit, 3-10
 Virtual I/O Bits, 3-14
 Virtual I/O Number, 3-11
 Virtual I/O Point Parameter, 3-14
 Virtual I/O Point Range, 3-14
 Virtual I/O Points, 3-15
 Virtual I/O vs. M-Functions, 3-1, 3-15

Virtual I/O vs. PLC I/O, 3-10
 Virtual I/O vs. Xycom I/O, 3-1, 3-14
 VME Address Parameter, 3-13
 VME Backplane, 3-13

W

WAIT command, 4-51
 Wait for Cycle Start, 3-4
 Warranty Policy, 2-1
 WATCH statement, 4-52
 WHILE, 4-24
 While Loop, 4-25
 While Loop, Nested, 4-25
 While Loops, 3-4
 While-Do-Endwhile, 4-24
 Window, Item Placing, 4-29
 WTCH statement, 4-52

X

XOR, 4-14
 XPlane Axis, 2-22
 XYCOM, 3-13
 XYCOM Base Address, 3-13
 XYCOM Card Number, 3-14
 Xycom Digital I/O Bits, 3-14
 XYCOM Digital I/O Cards, 3-1, 3-13
 Xycom I/O vs. Virtual I/O, 3-1, 3-14

Y

YPlane Axis, 2-22



READER'S COMMENTS

UNIDEX U600 Series Programming Manual P/N EDU 152, December 1996

Please answer the questions below and add any suggestions for improving this document. Is the information:

	Yes	No
Adequate to the subject?	___	___
Well organized?	___	___
Clearly presented?	___	___
Well illustrated?	___	___
Would you like to see more illustrations?	___	___
Would you like to see more text?	___	___

How do you use this document in your job? Does it meet your needs?

What improvements, if any, would you like to see? Please be specific or cite examples.

Your name _____
Your title _____
Company name _____
Address _____

Remove this page from the document and fax or mail your comments to the technical writing department of Aerotech.

AEROTECH, INC.
Technical Writing Department
101 Zeta Drive
Pittsburgh, PA. 15238-2897 U.S.A.
Fax number (412) 963-7009

